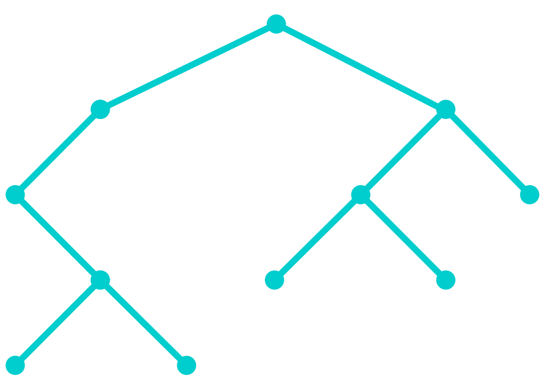
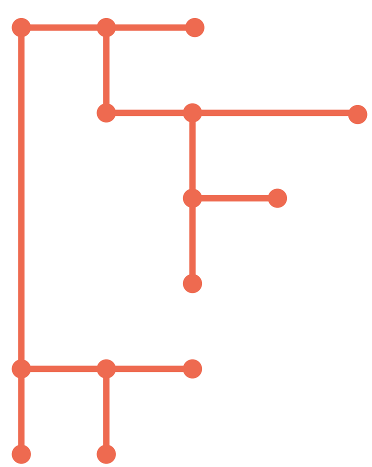
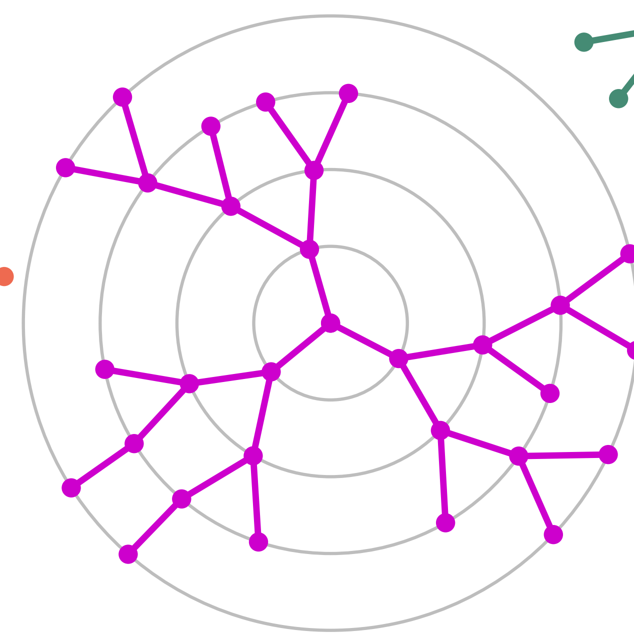
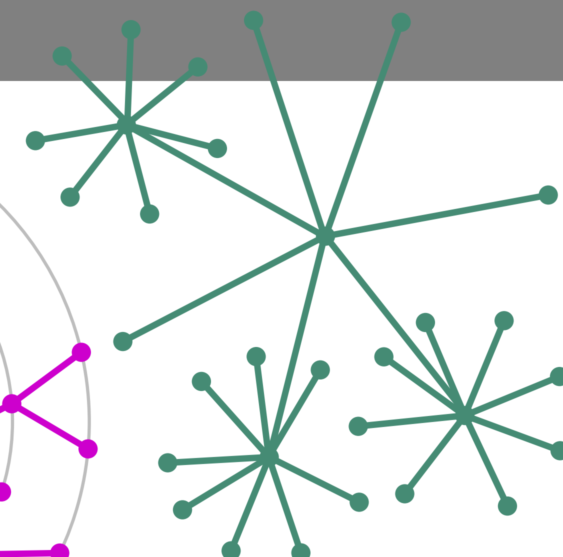
Algorithms for graph visualization

Divide and Conquer - Tree Layouts

WINTER SEMESTER 2017/2018

Tamara Mchedlidze

KIT – Universität des Landes Baden-Württemberg und
nationales Forschungszentrum in der Helmholtz-Gemeinschaft

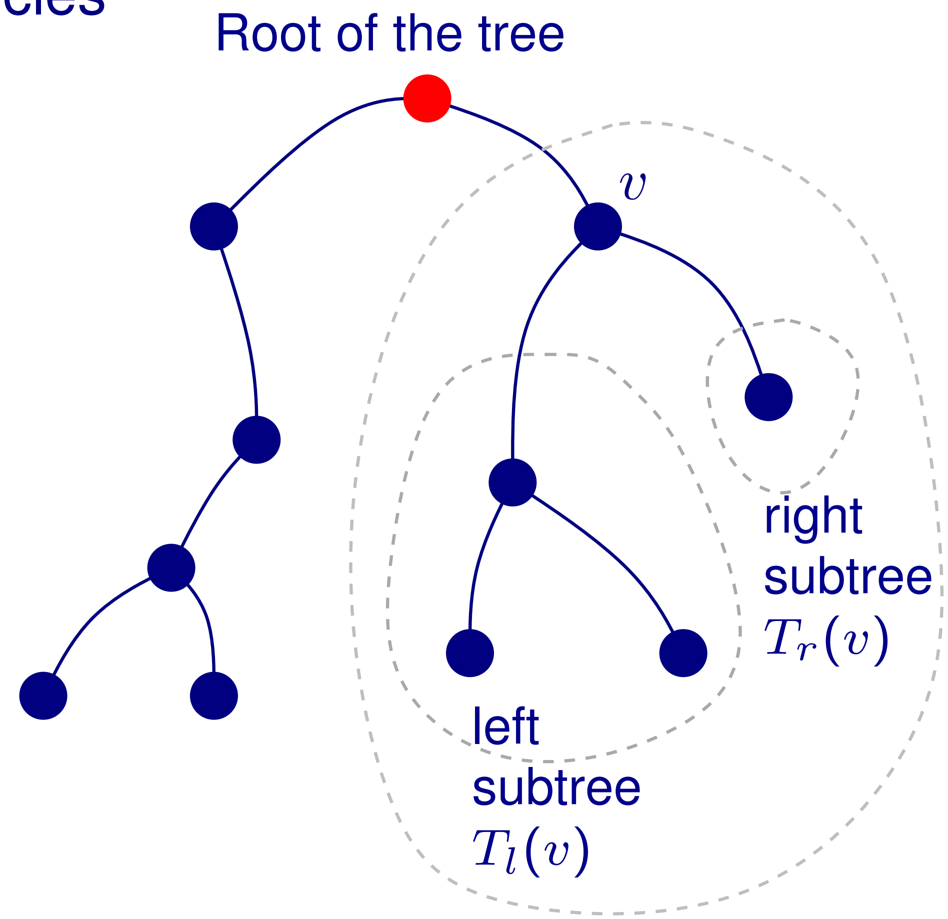
www.kit.edu

Overview

- Basic Definitions
- Level-based tree layout algorithm
- H(horizontal) V(vertical) tree layout algorithm
- Radial tree layout algorithm
- Other visualization styles

Basic Definitions

- Tree - connected graph without cycles
- Binary tree

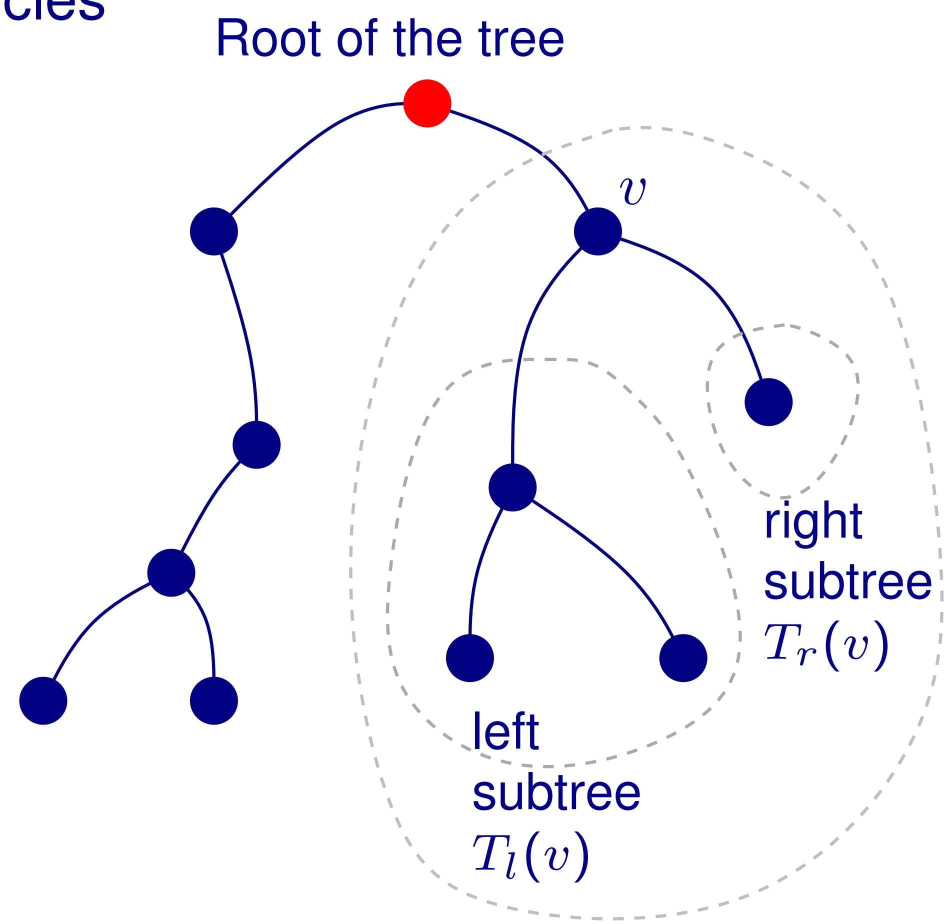


3 - 1

Basic Definitions

- Tree - connected graph without cycles
- Binary tree

Tree traversals



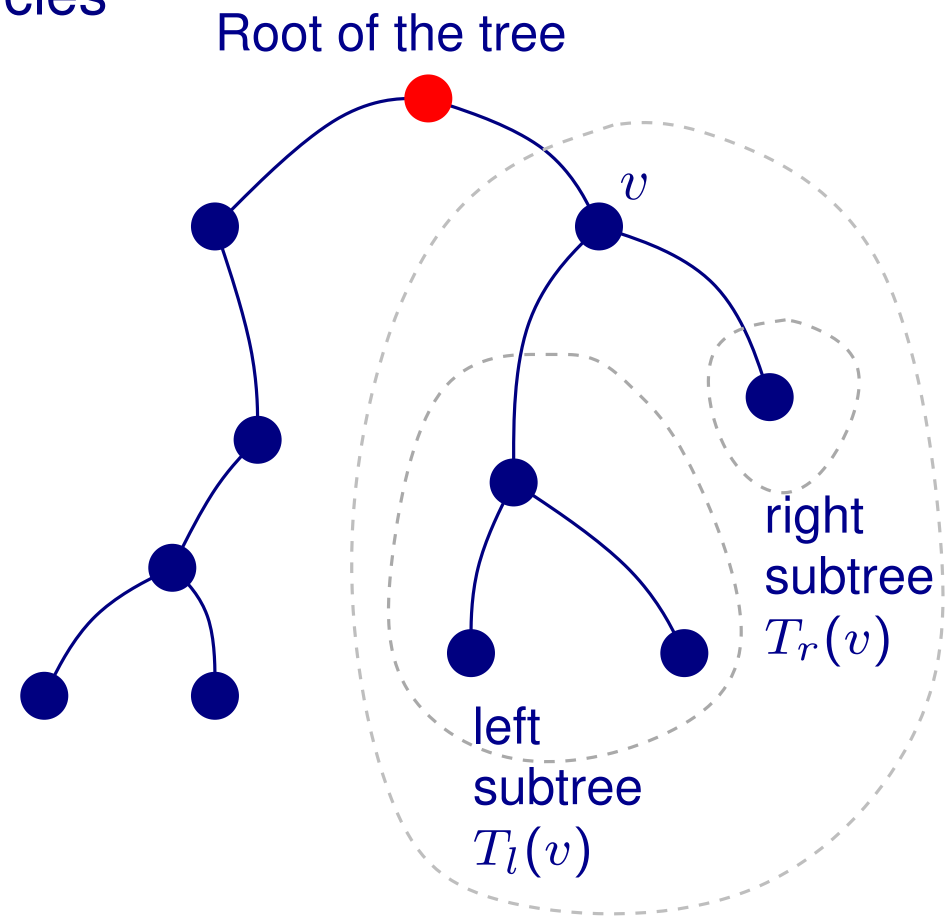
3 - 2

Basic Definitions

- Tree - connected graph without cycles
- Binary tree

Tree traversals

Depth-first search

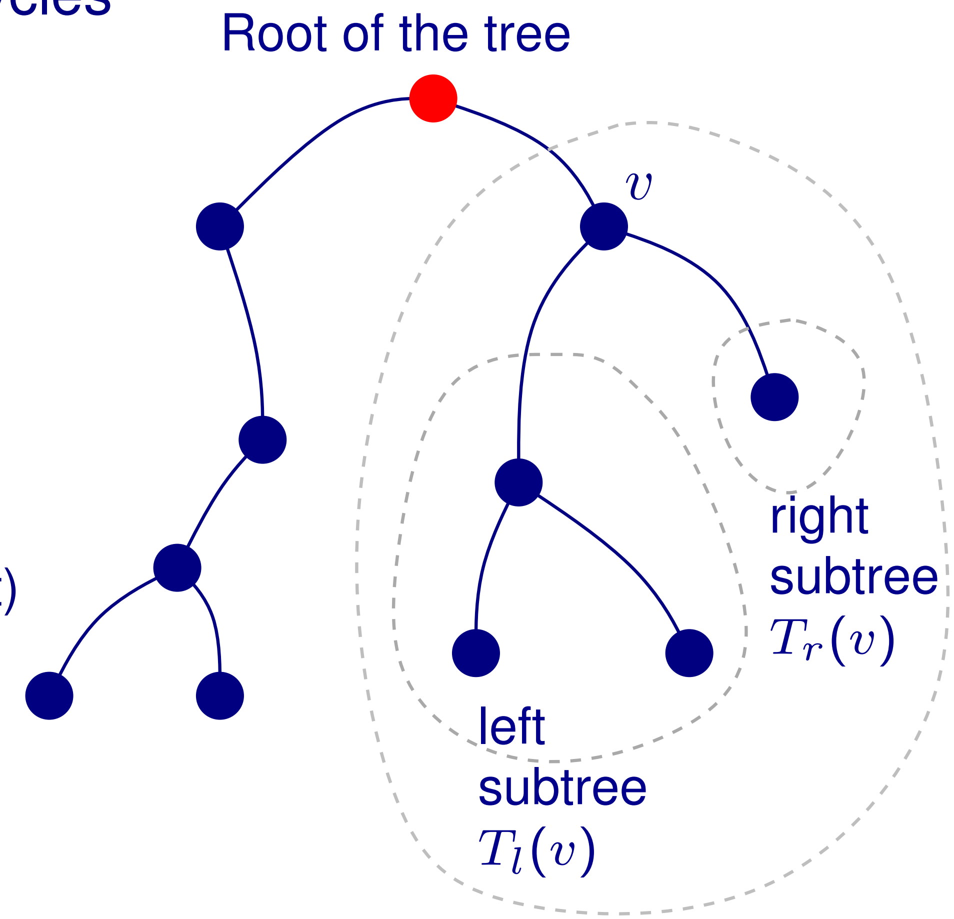


- Tree - connected graph without cycles
- Binary tree

Tree traversals

Depth-first search

- Pre-order (First parent, then subtrees)
- In-order (Left child, parent, right child)
- Post-order (First subtrees, then parent)



Basic Definitions

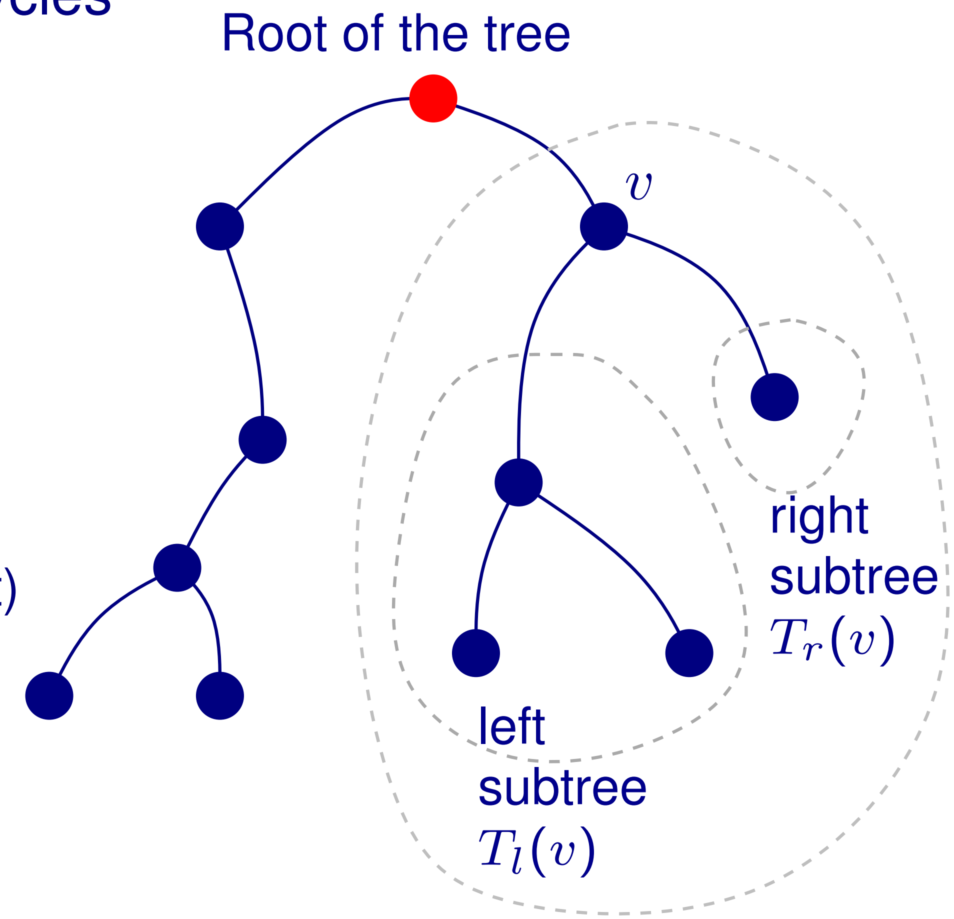
- Tree - connected graph without cycles
- Binary tree

Tree traversals

Depth-first search

- Pre-order (First parent, then subtrees)
- In-order (Left child, parent, right child)
- Post-order (First subtrees, then parent)

Breadth-first search



Basic Definitions

- Tree - connected graph without cycles
- Binary tree

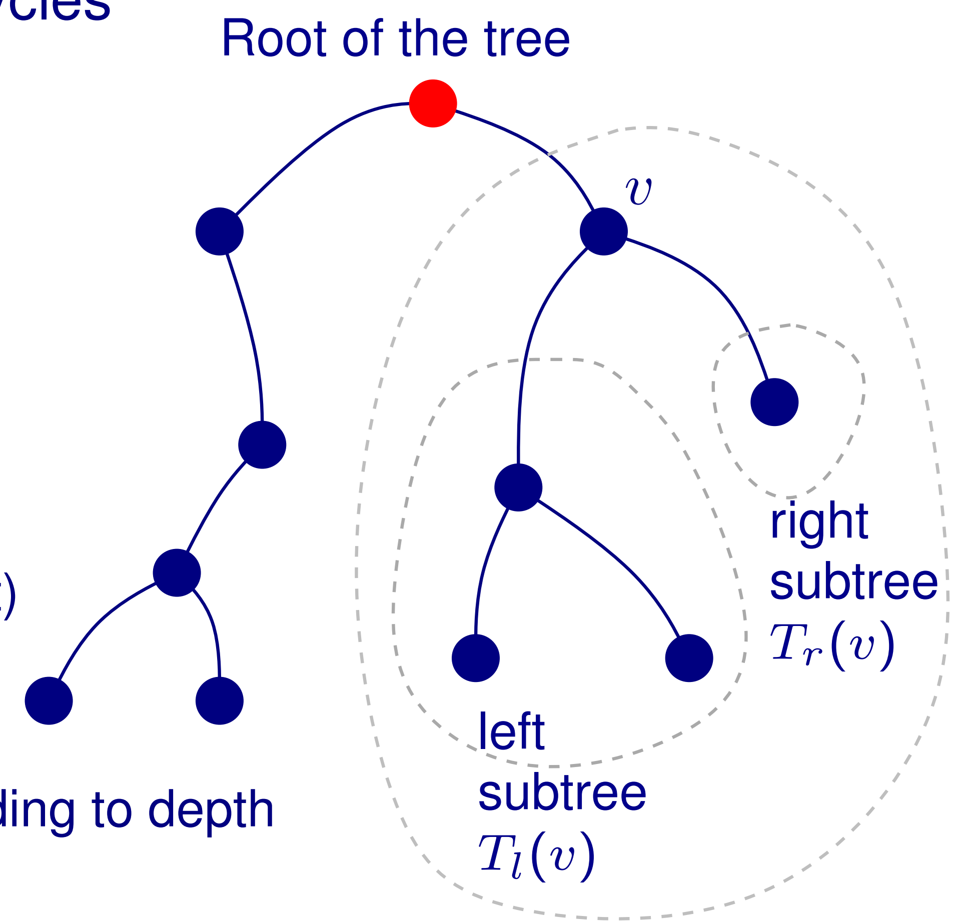
Tree traversals

Depth-first search

- Pre-order (First parent, then subtrees)
- In-order (Left child, parent, right child)
- Post-order (First subtrees, then parent)

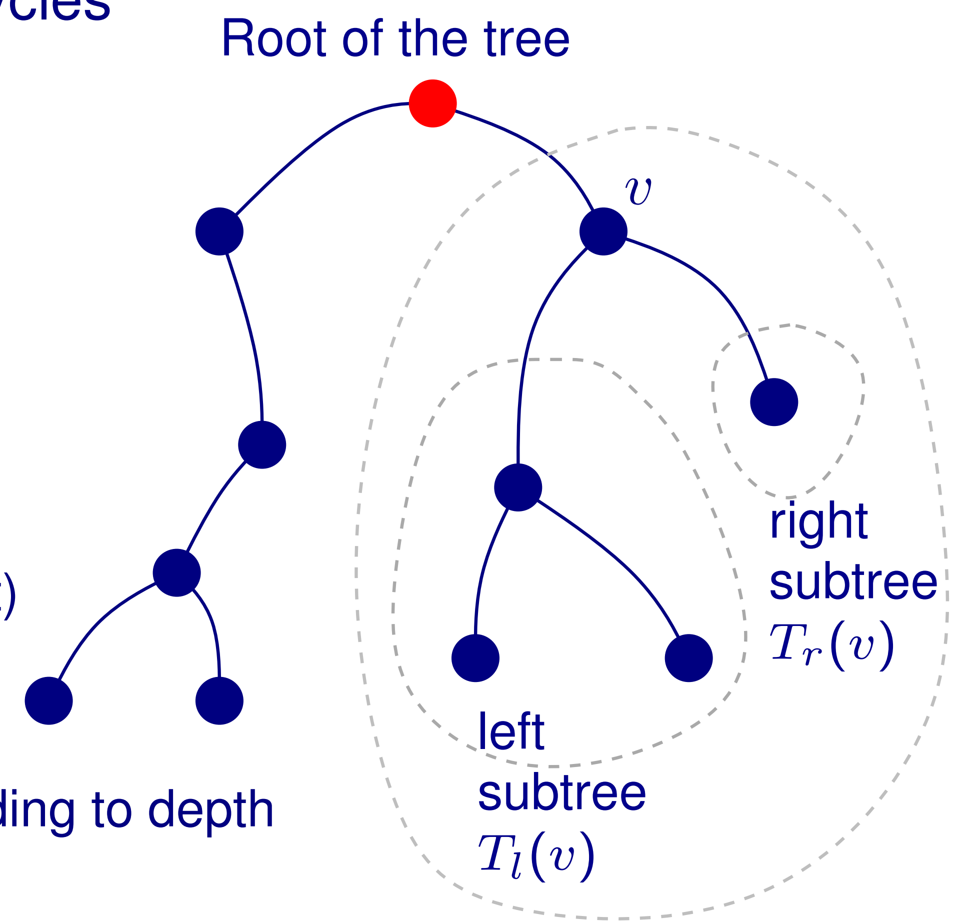
Breadth-first search

- Assignes vertices to levels corresponding to depth



Basic Definitions

- Tree - connected graph without cycles
- Binary tree



Tree traversals

Depth-first search

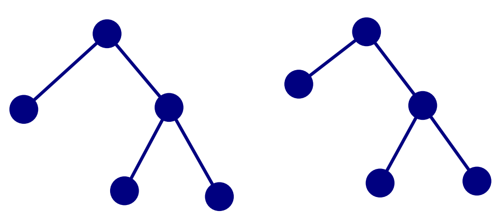
- Pre-order (First parent, then subtrees)
- In-order (Left child, parent, right child)
- Post-order (First subtrees, then parent)

Breadth-first search

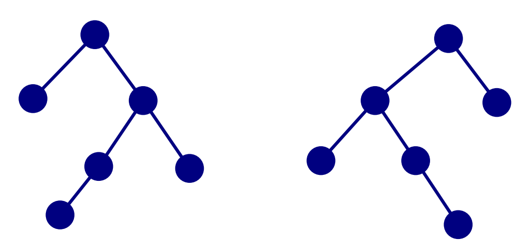
- Assignes vertices to levels corresponding to depth

Isomorphism (of ordered trees)

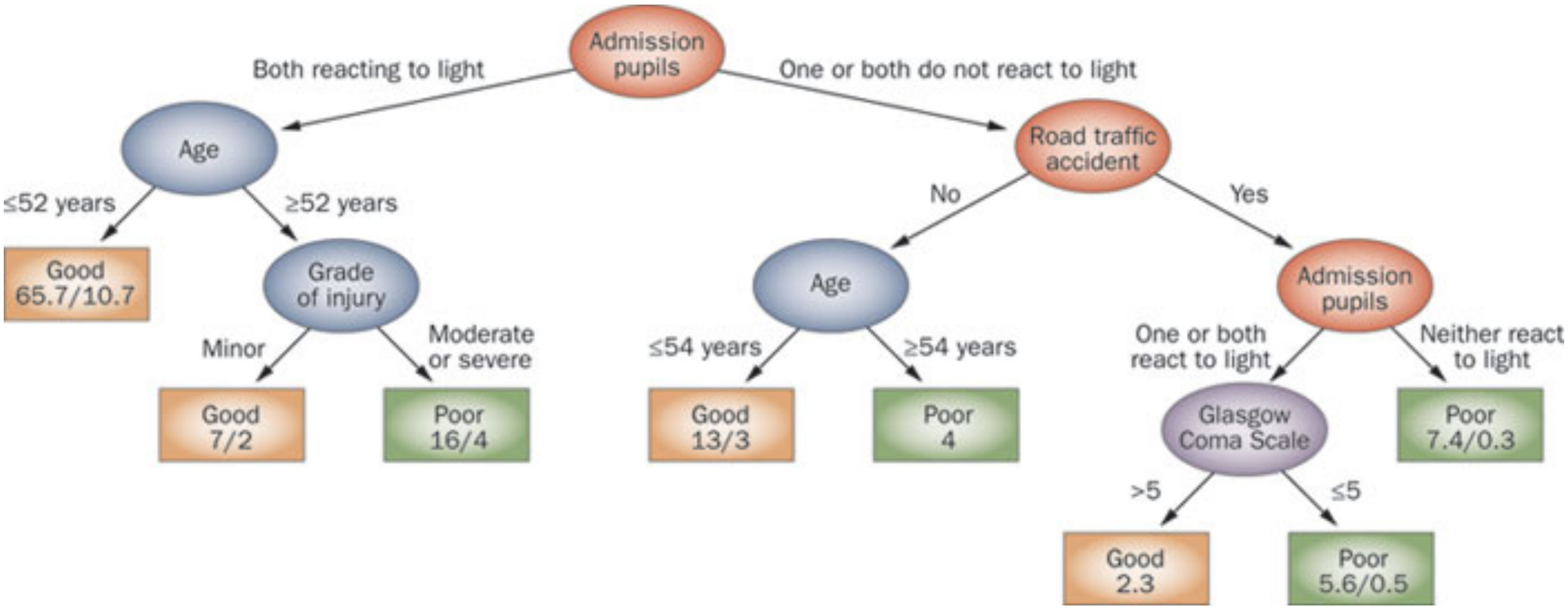
Simple



Axial



Applications



Decision tree analysis for prediction of outcome after traumatic brain injury
Nature Reviews Neurology

Applications

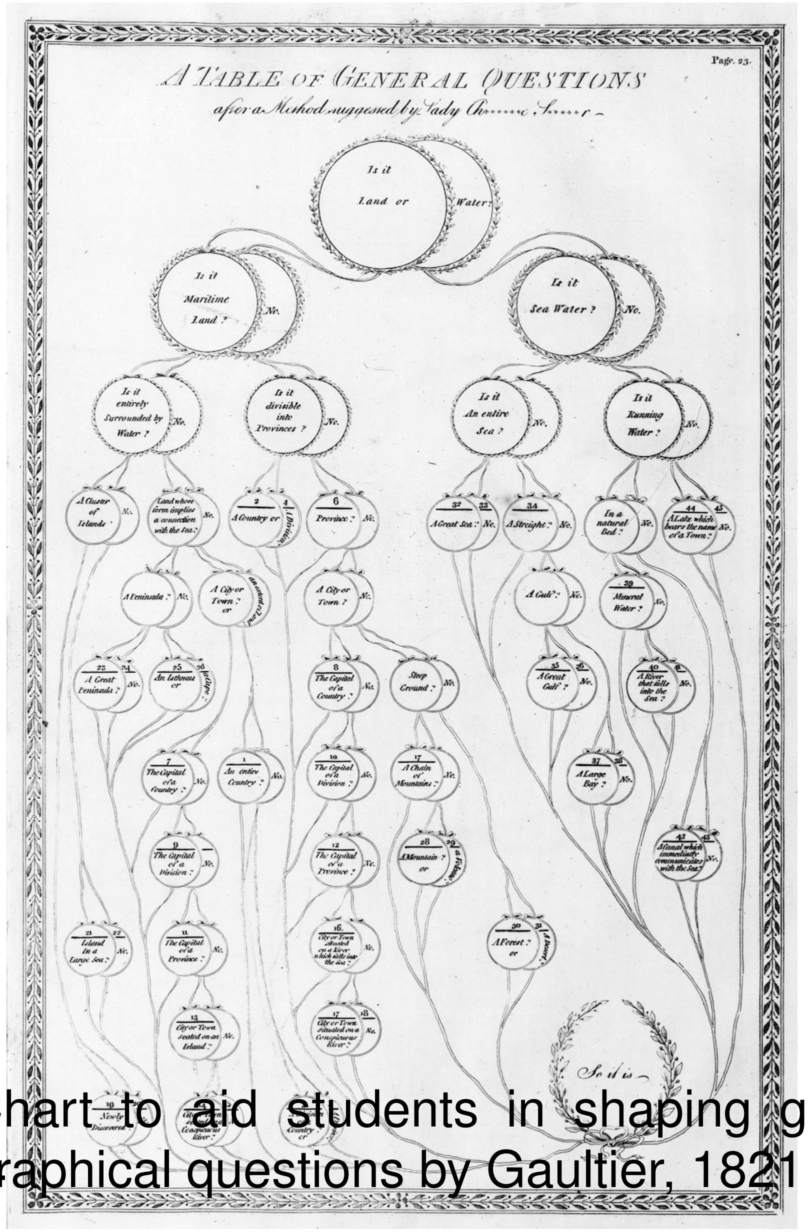


Chart to aid students in shaping geographical questions by Gaultier, 1821

Applications

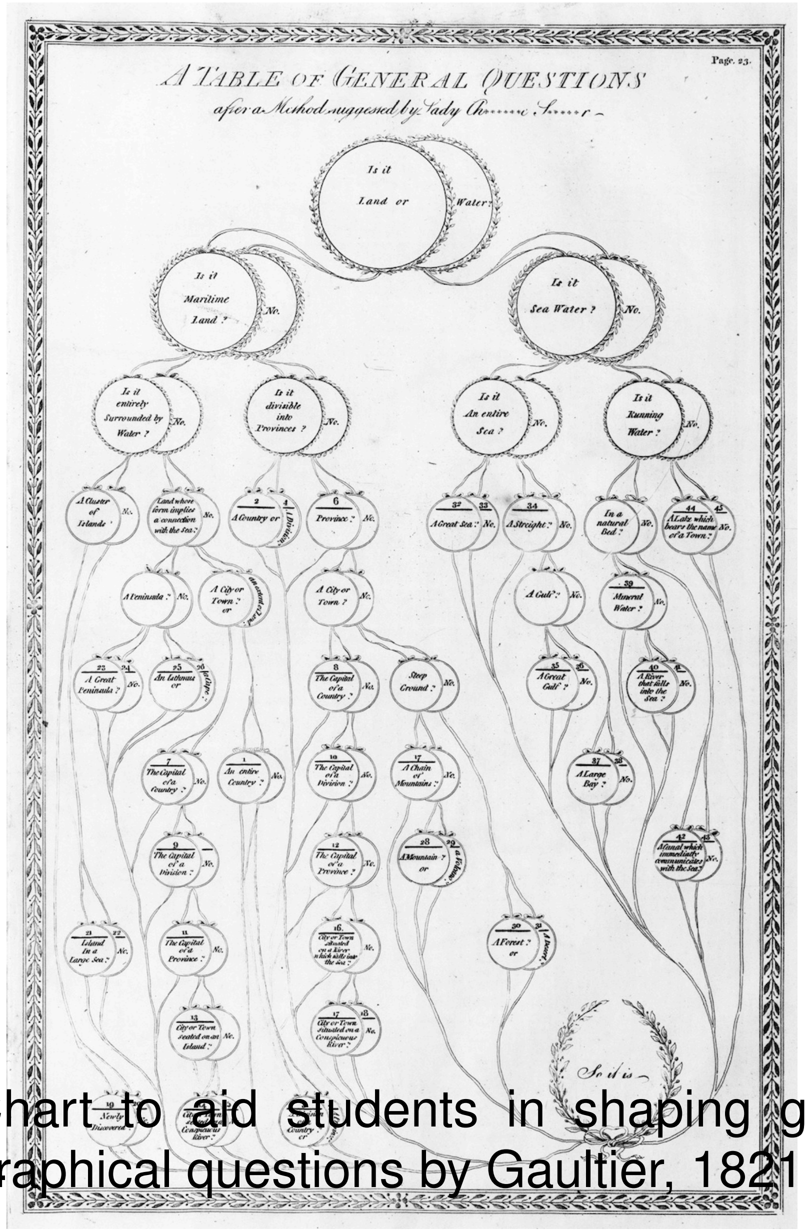
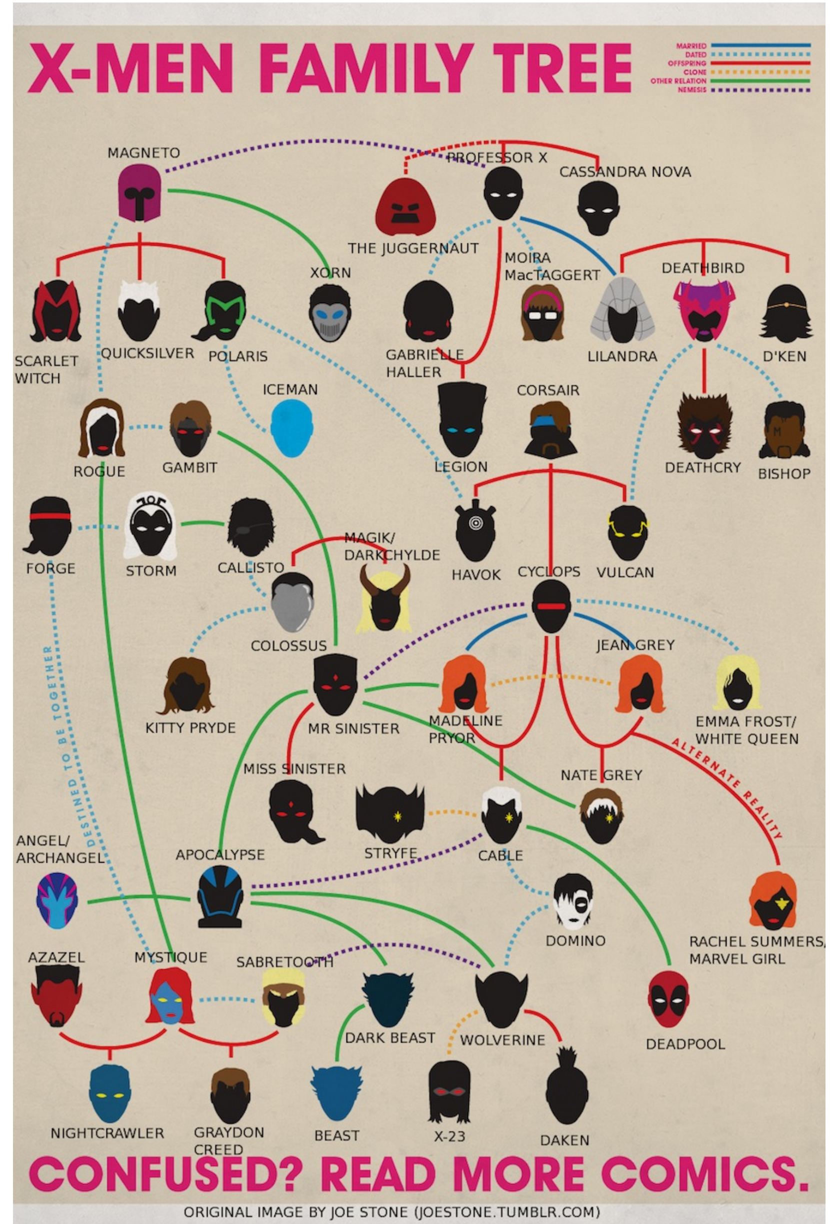


Chart to aid students in shaping geographical questions by Gaultier, 1821

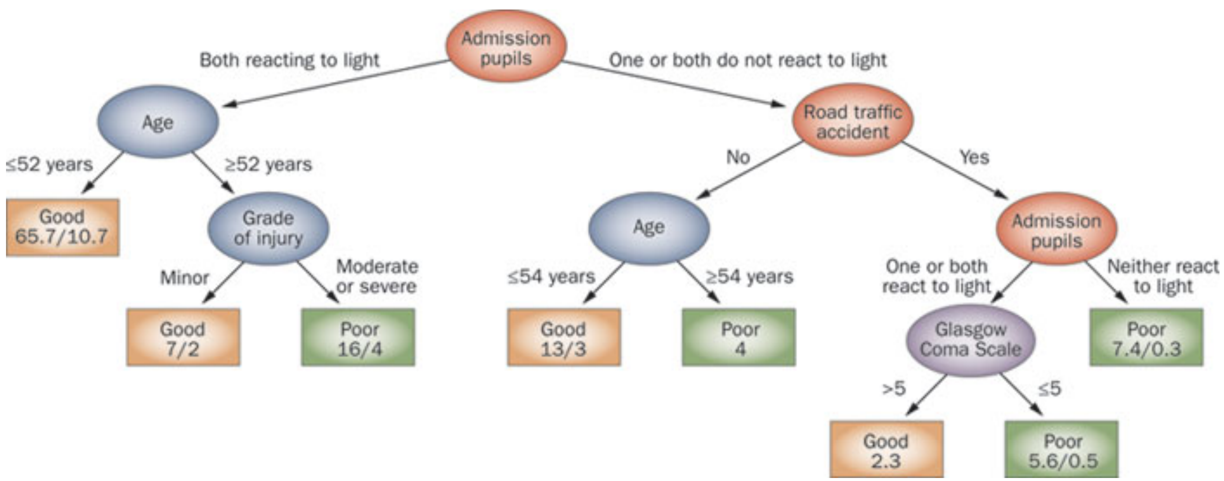


Level-Based Layout of a Tree

Discuss with your neighbour or in groups of three and write down

5 min

- What the properties of the layout?
- What are the drawing conventions and the aesthetics that we have to take into account?

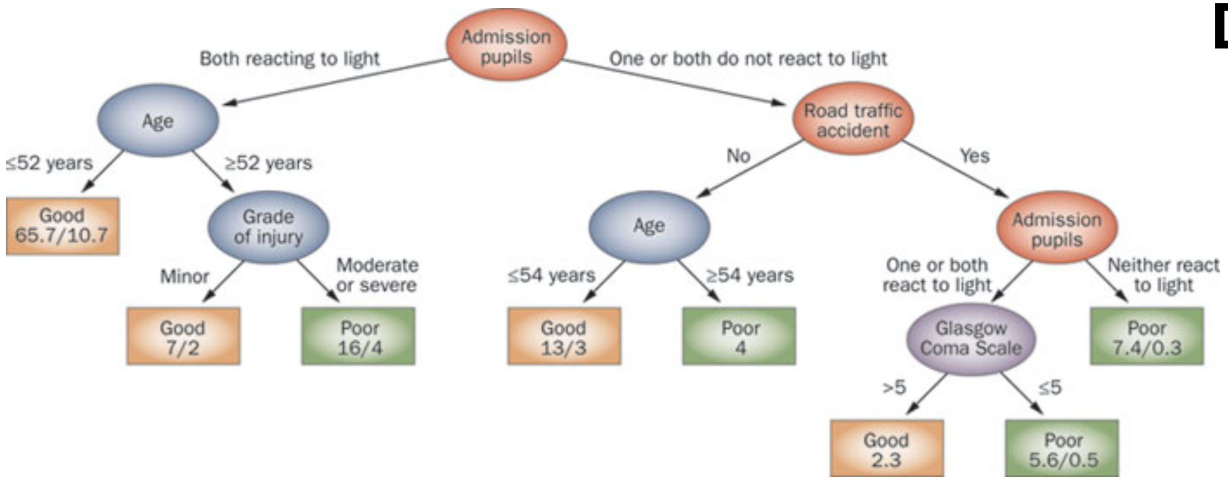


Level-Based Layout of a Tree

Discuss with your neighbour or in groups of three and write down

5 min

- What the properties of the layout?
- What are the drawing conventions and the aesthetics that we have to take into account?



Drawing Conventions

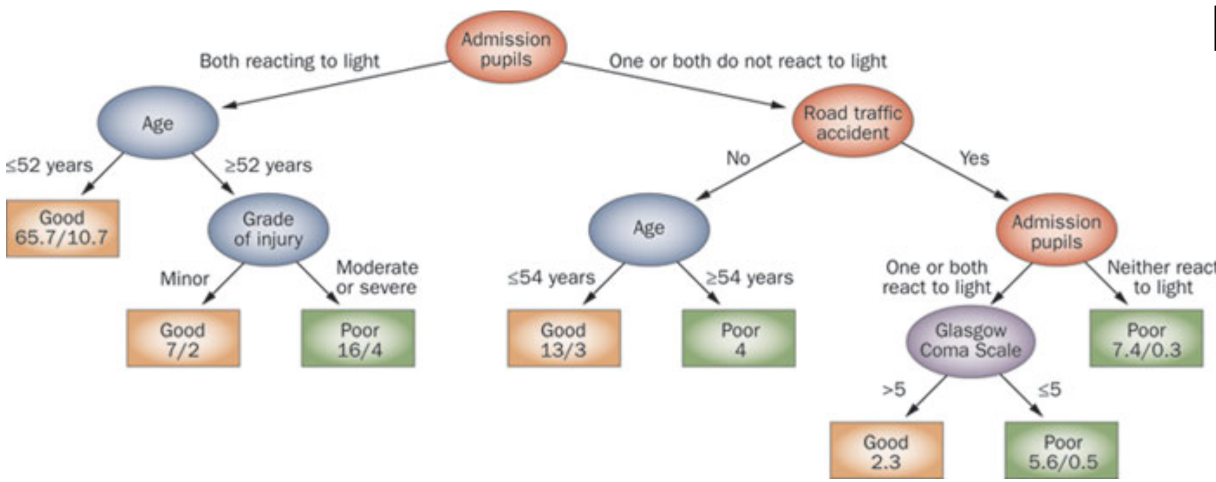
- Vertices lie on layers
- Parent is above the children
- Edges are straight lines
- Parent is centred with respect to the children
- Isomorphic subtrees have identical drawings

Level-Based Layout of a Tree

Discuss with your neighbour or in groups of three and write down

5 min

- What the properties of the layout?
- What are the drawing conventions and the aesthetics that we have to take into account?



Drawing Conventions

- Vertices lie on layers
- Parent is above the children
- Edges are straight lines
- Parent is centred with respect to the children
- Isomorphic subtrees have identical drawings

Drawing Aesthetics

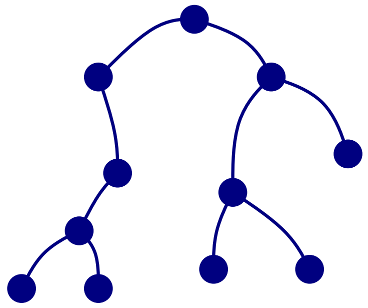
- Area

Level-based Layout

Algorithm Outline:

Input: A binary tree

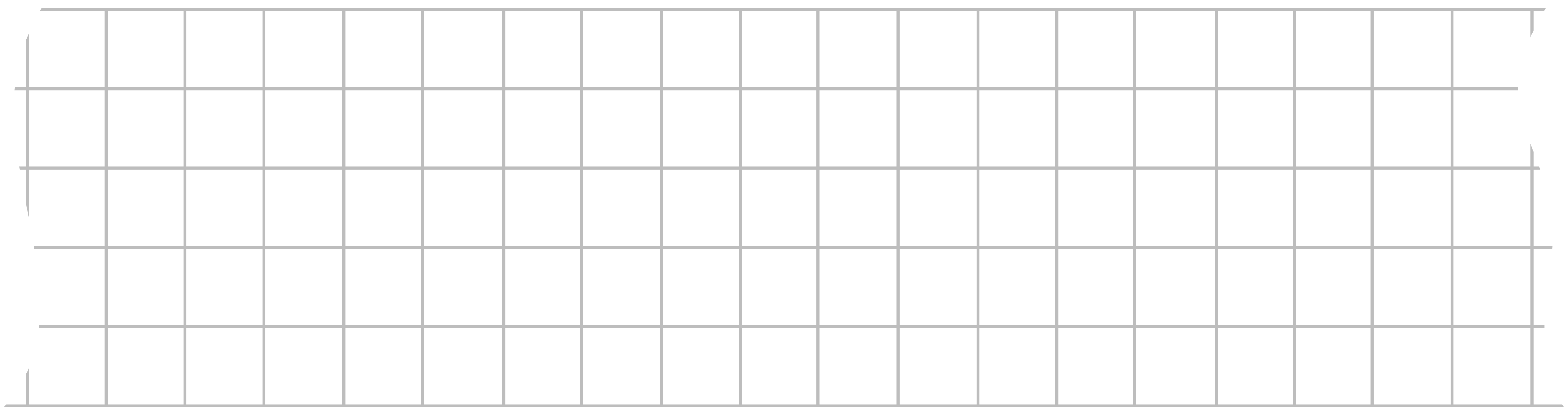
Output: A leveled drawing of T



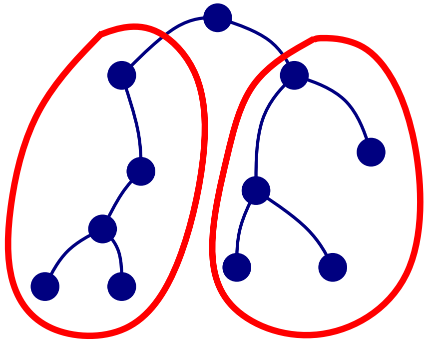
Base case: A single vertex

Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:



Level-based Layout



Algorithm Outline:

Input: A binary tree

Output: A leveled drawing of T

Base case: A single vertex

Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

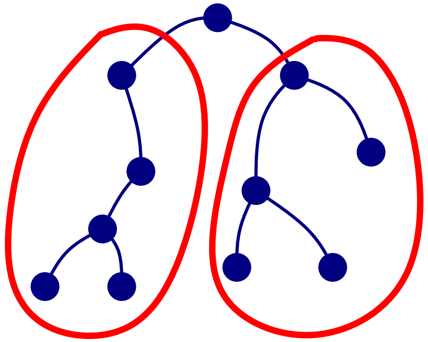
Conquer:

Level-based Layout

Algorithm Outline:

Input: A binary tree

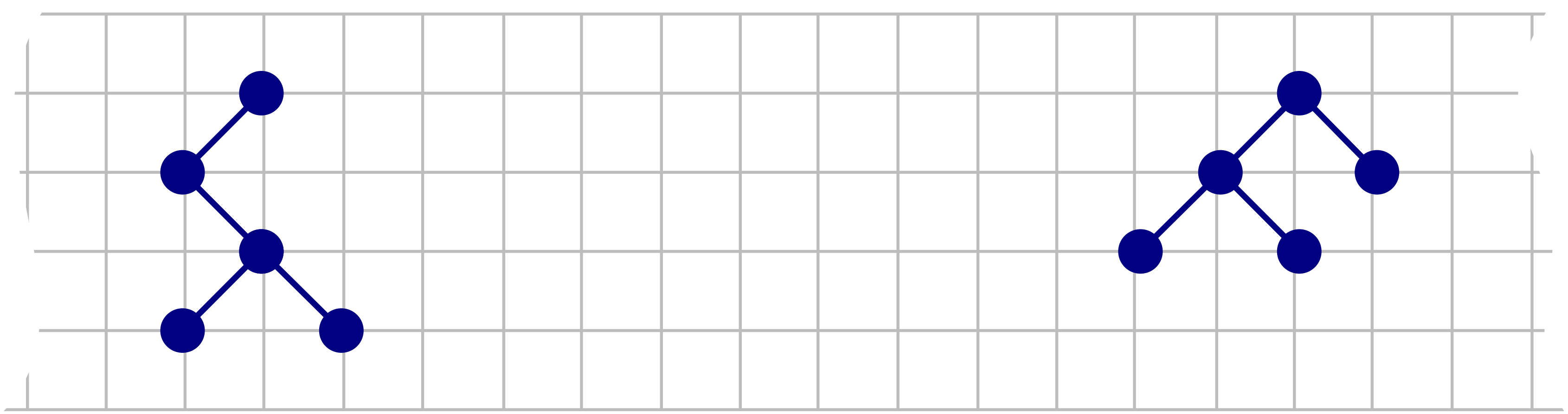
Output: A leveled drawing of T



Base case: A single vertex

Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:

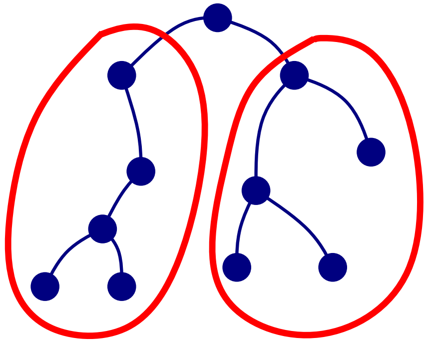


Level-based Layout

Algorithm Outline:

Input: A binary tree

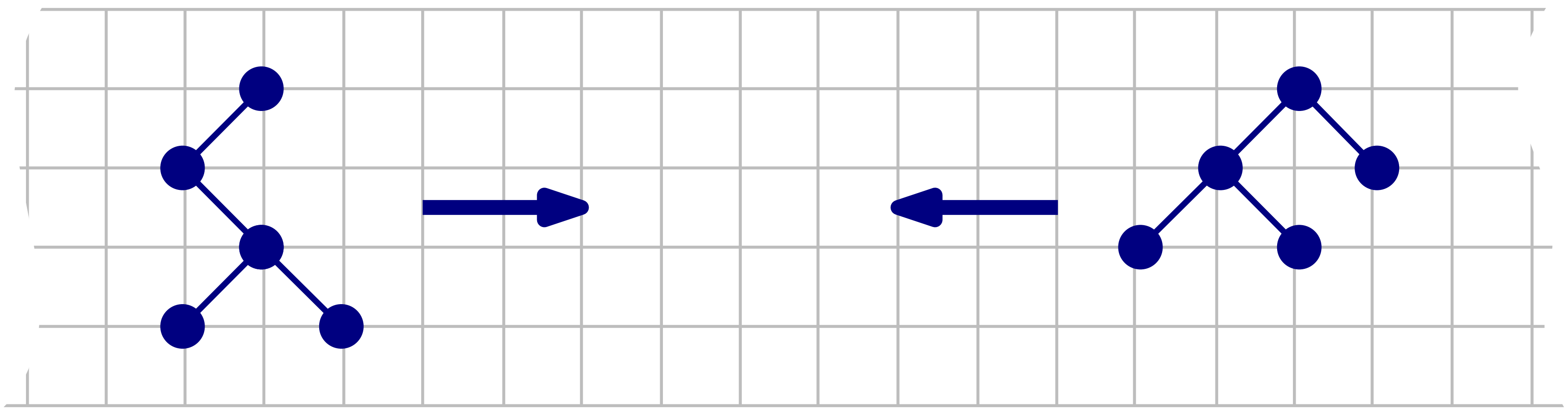
Output: A leveled drawing of T



Base case: A single vertex

Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:

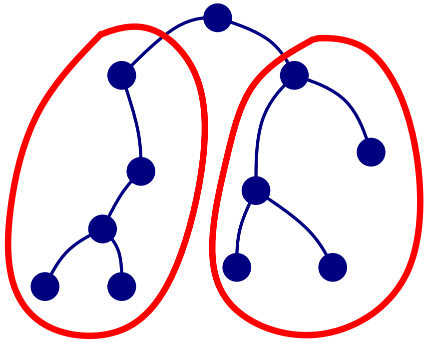


Level-based Layout

Algorithm Outline:

Input: A binary tree

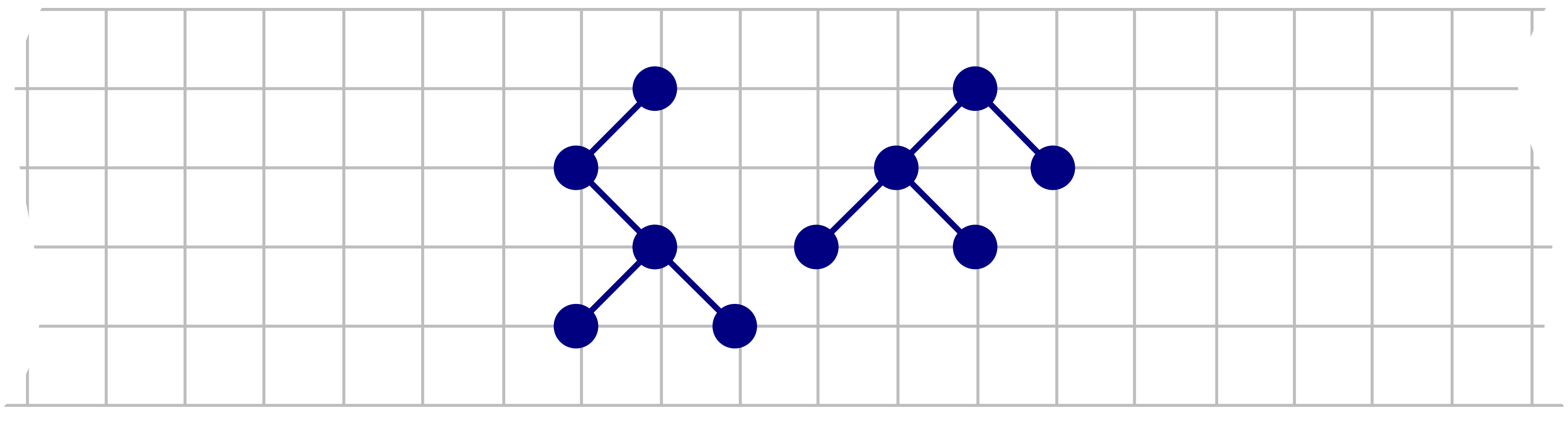
Output: A leveled drawing of T



Base case: A single vertex

Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:

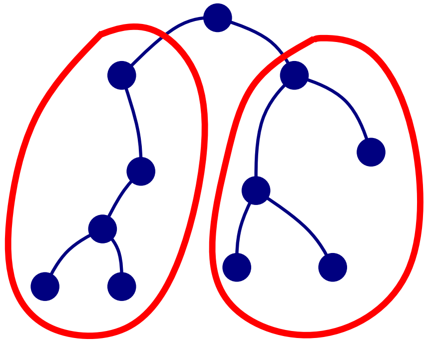


Level-based Layout

Algorithm Outline:

Input: A binary tree

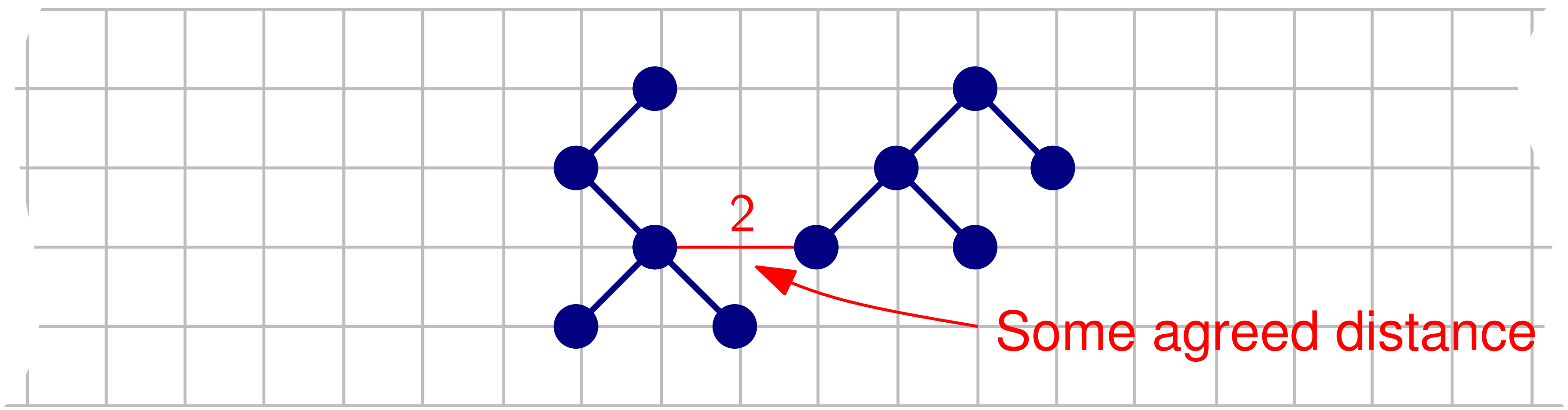
Output: A leveled drawing of T



Base case: A single vertex

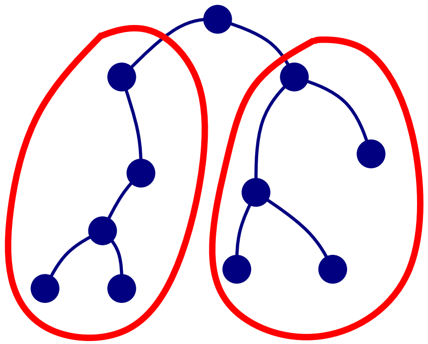
Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:



Level-based Layout

Algorithm Outline:



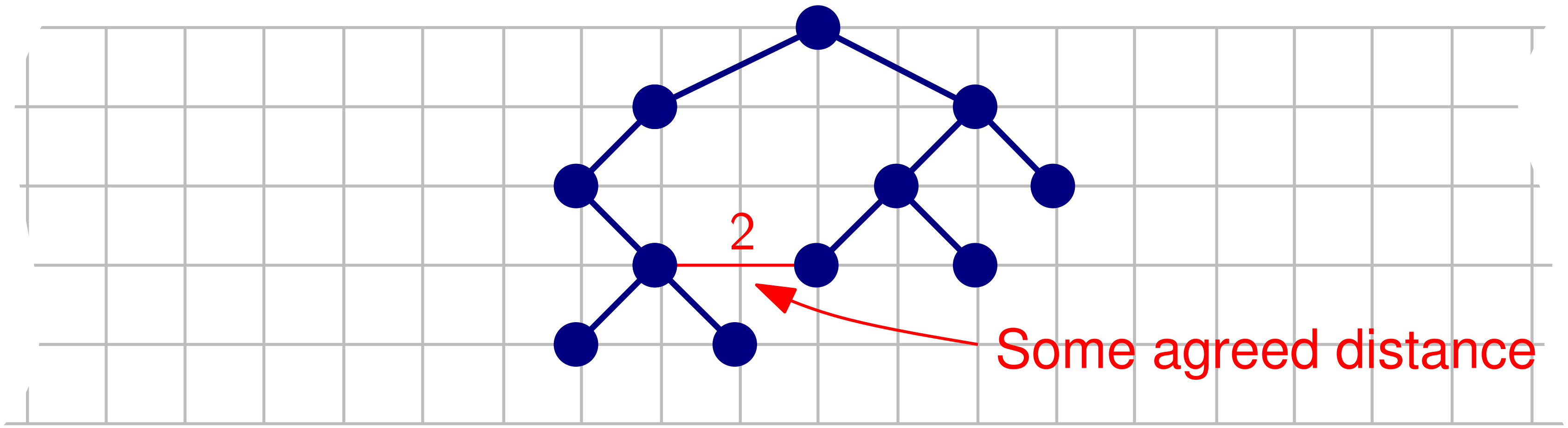
Input: A binary tree

Output: A leveled drawing of T

Base case: A single vertex

Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:

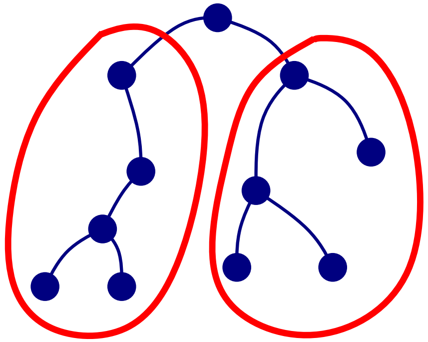


Level-based Layout

Algorithm Outline:

Input: A binary tree

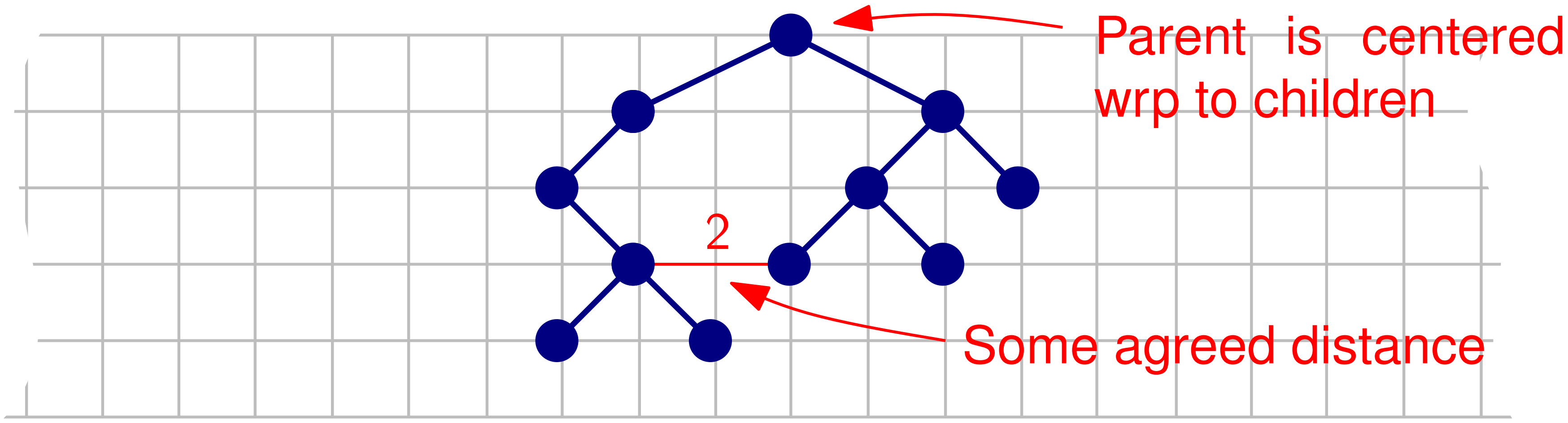
Output: A leveled drawing of T



Base case: A single vertex

Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

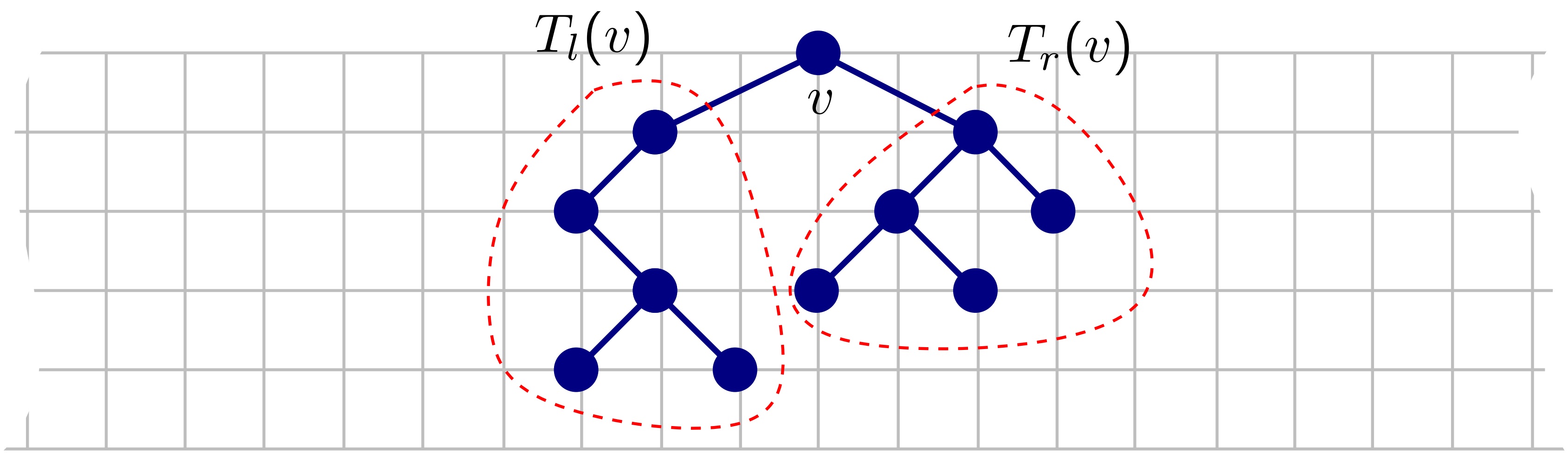
Conquer:



Level-based Layout

Implementation Details (postorder and preorder traversals)

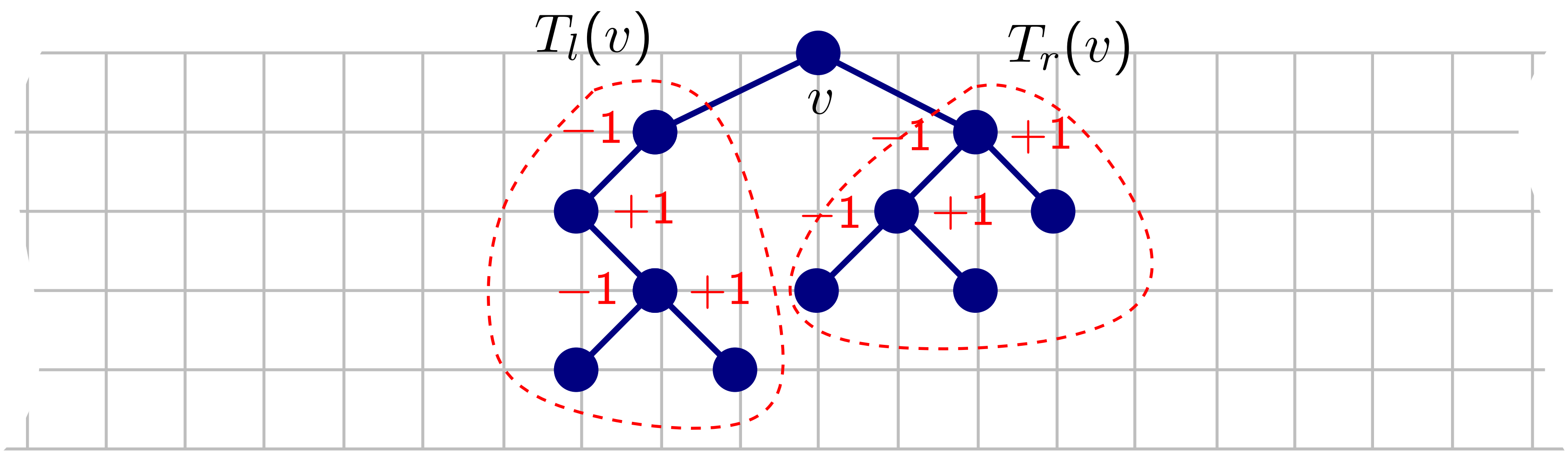
Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child



Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

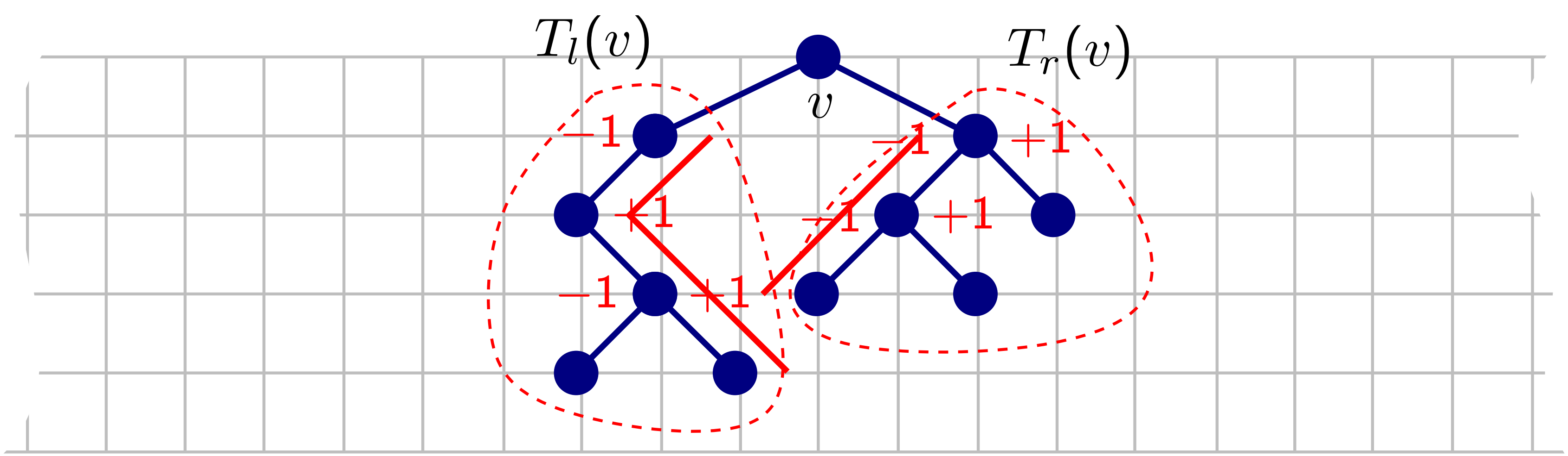


Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

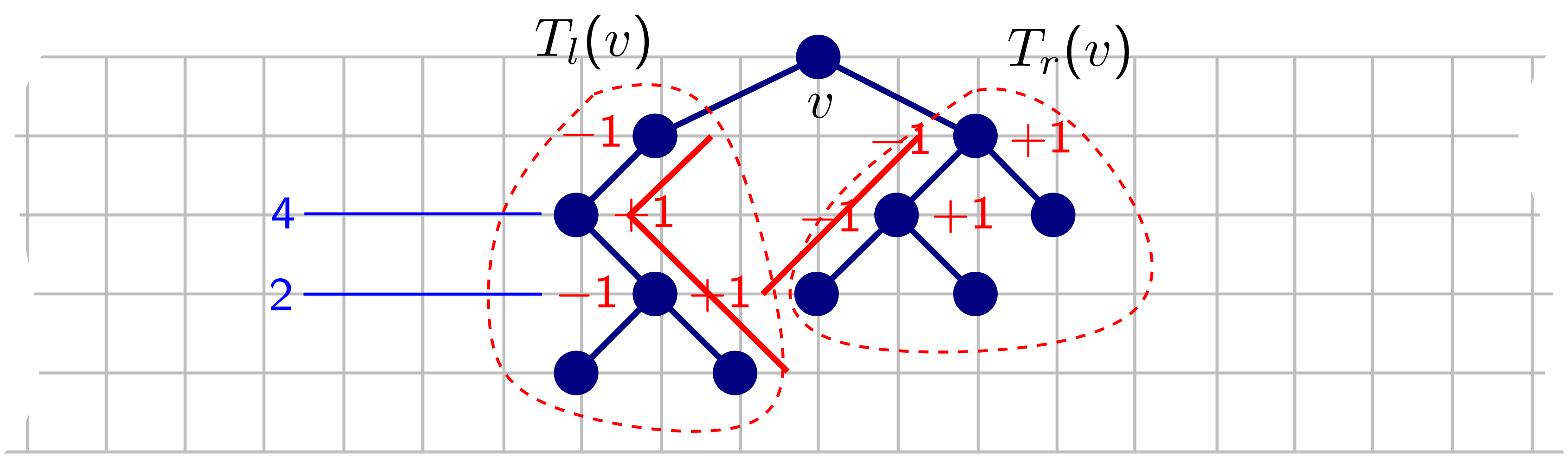
- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$



Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$



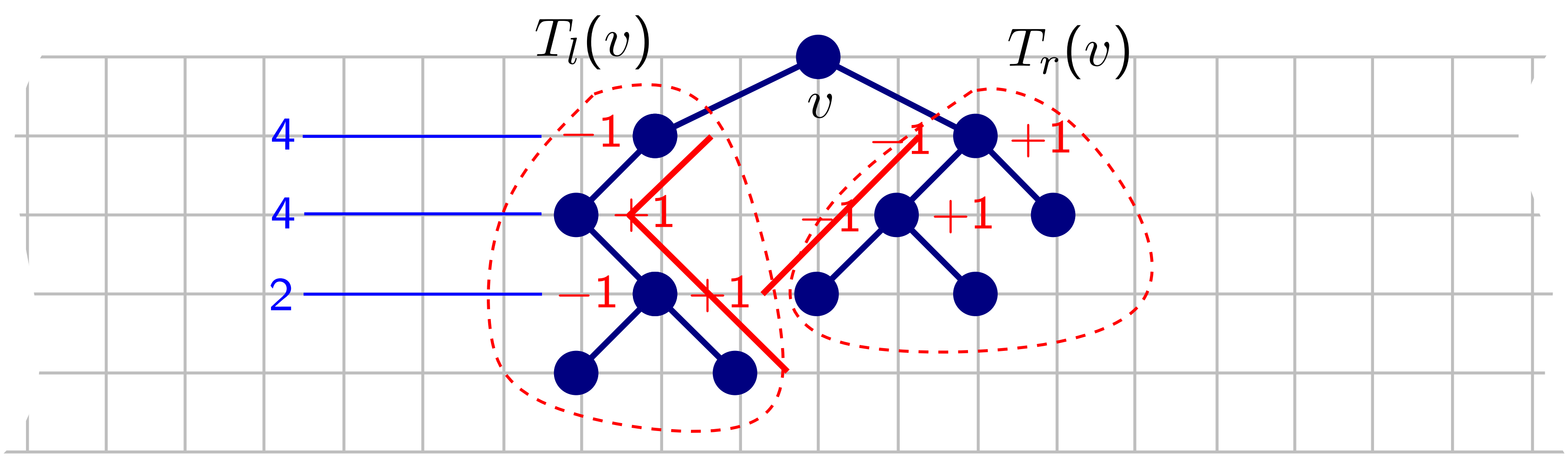
Level-based Layout

Karlsruhe Institute of Technology

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$

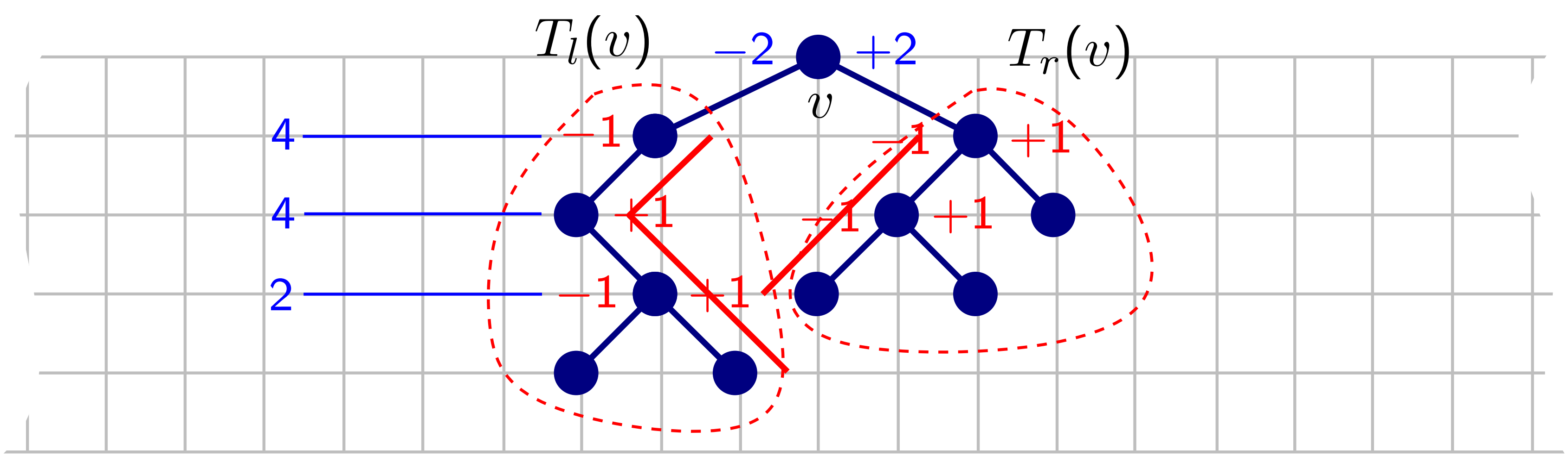


Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$

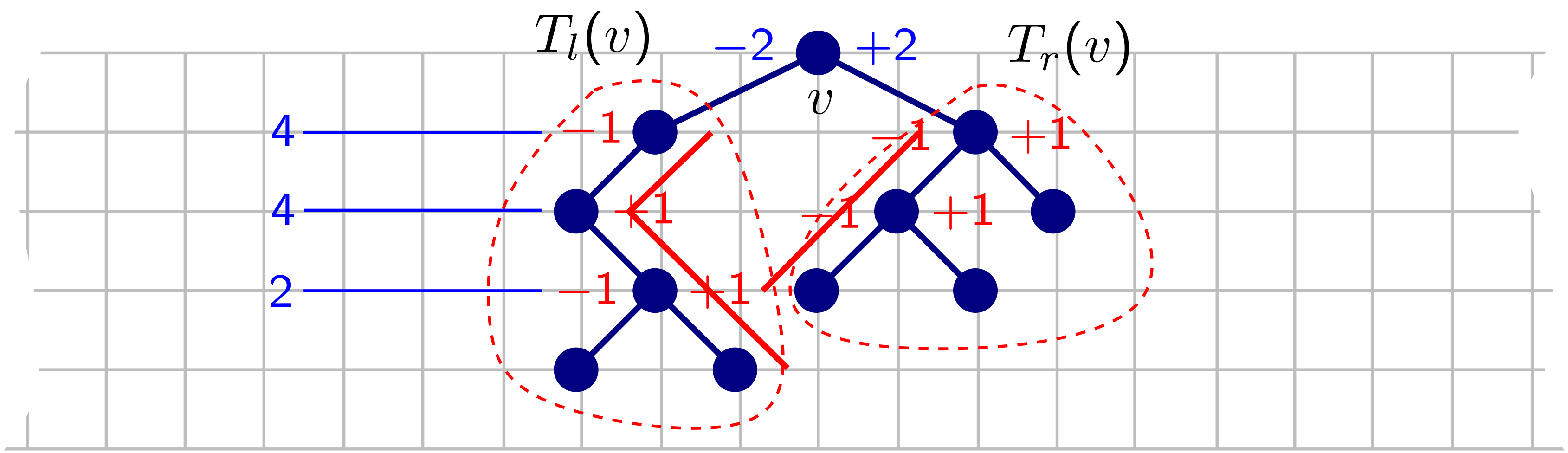


Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

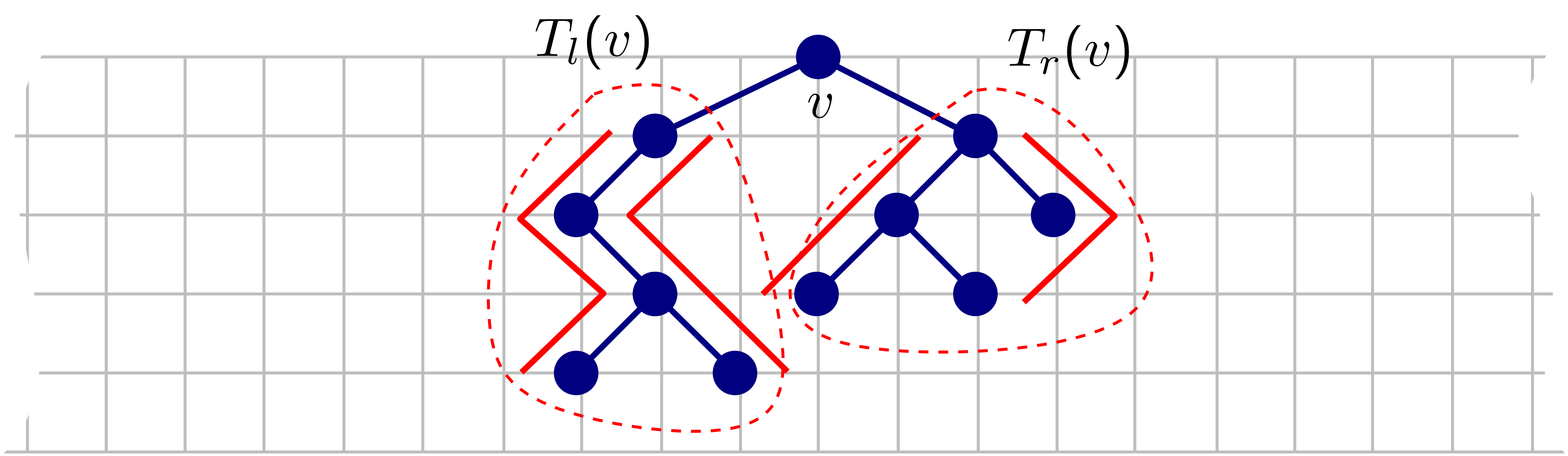
- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- "Summ up" the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$ to obtain the displ. of the children of v



Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- "Summ up" the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$ to obtain the displ. of the children of v

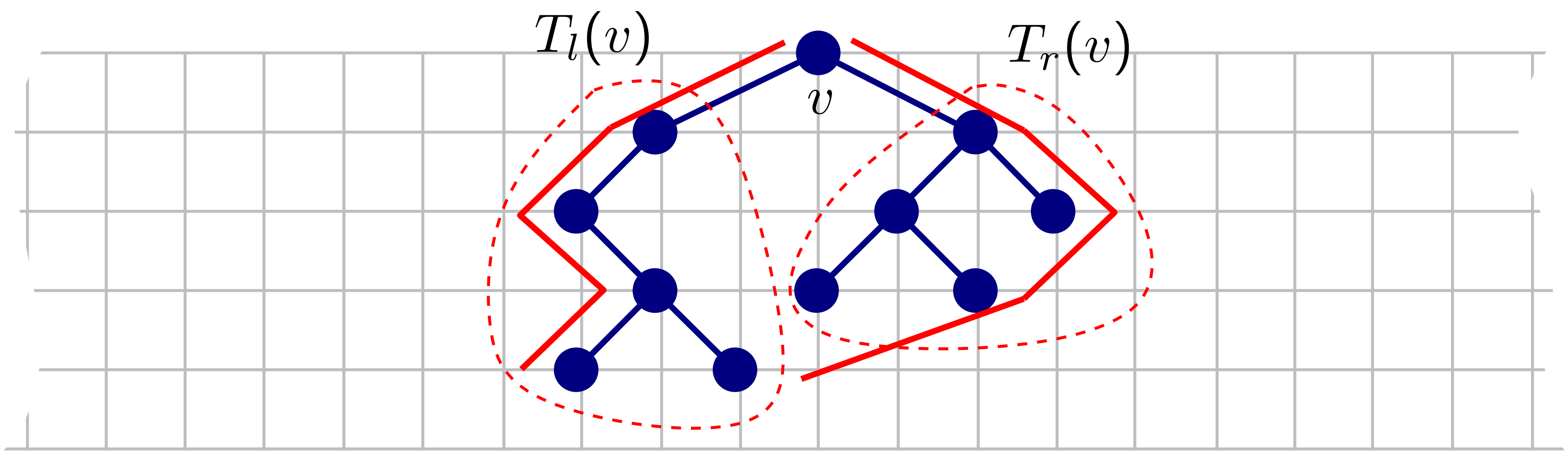


Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- "Summ up" the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$ to obtain the displ. of the children of v

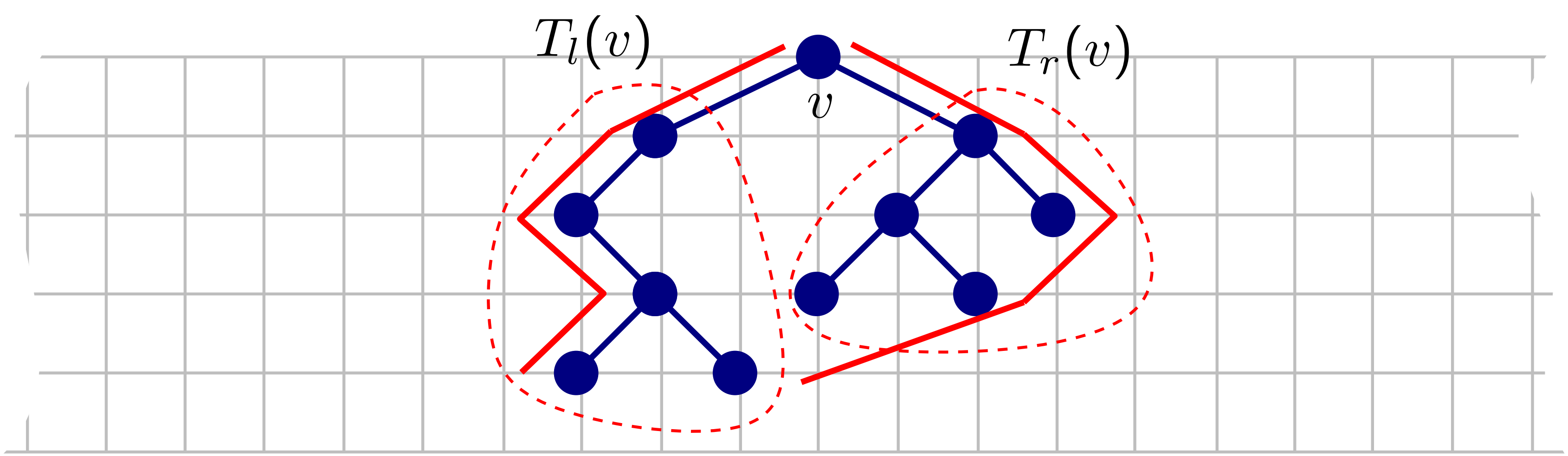


Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- "Summ up" the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$ to obtain the displ. of the children of v
- Store at v the left and the right boundaries of $T(v)$

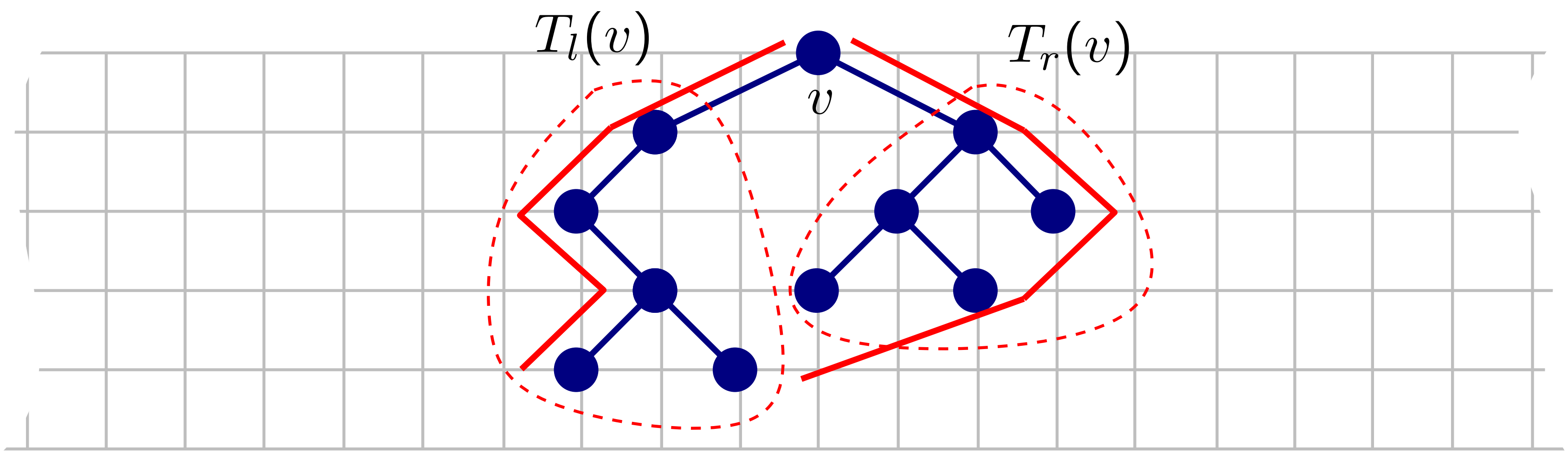


Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

Preorder traversal: Compute x- and y-coordinates.

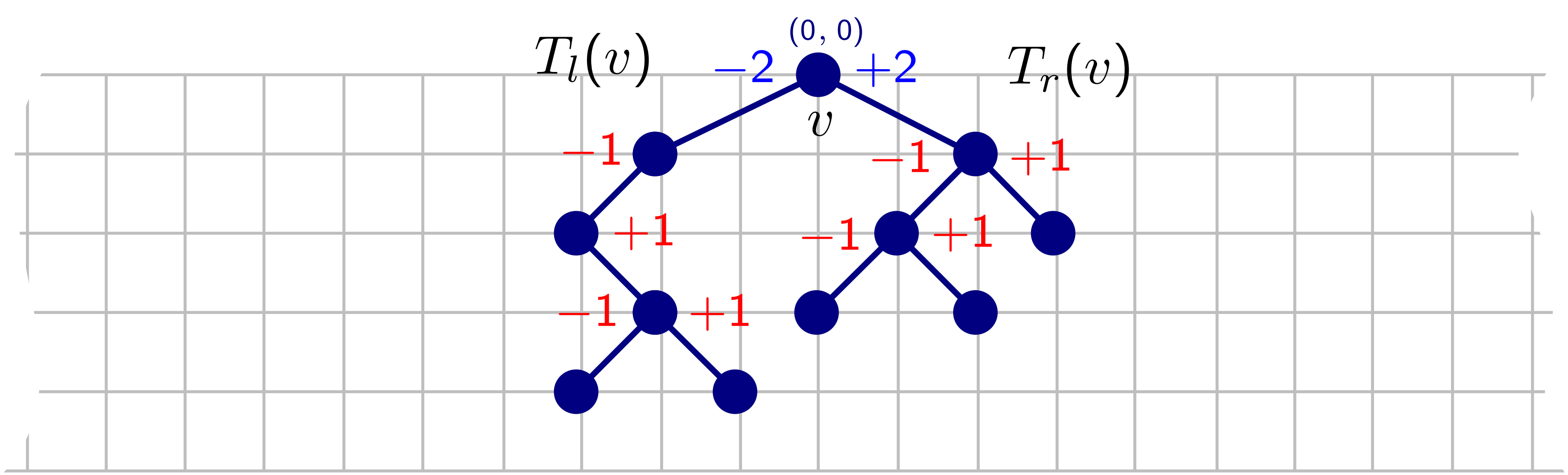


Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

Preorder traversal: Compute x- and y-coordinates.

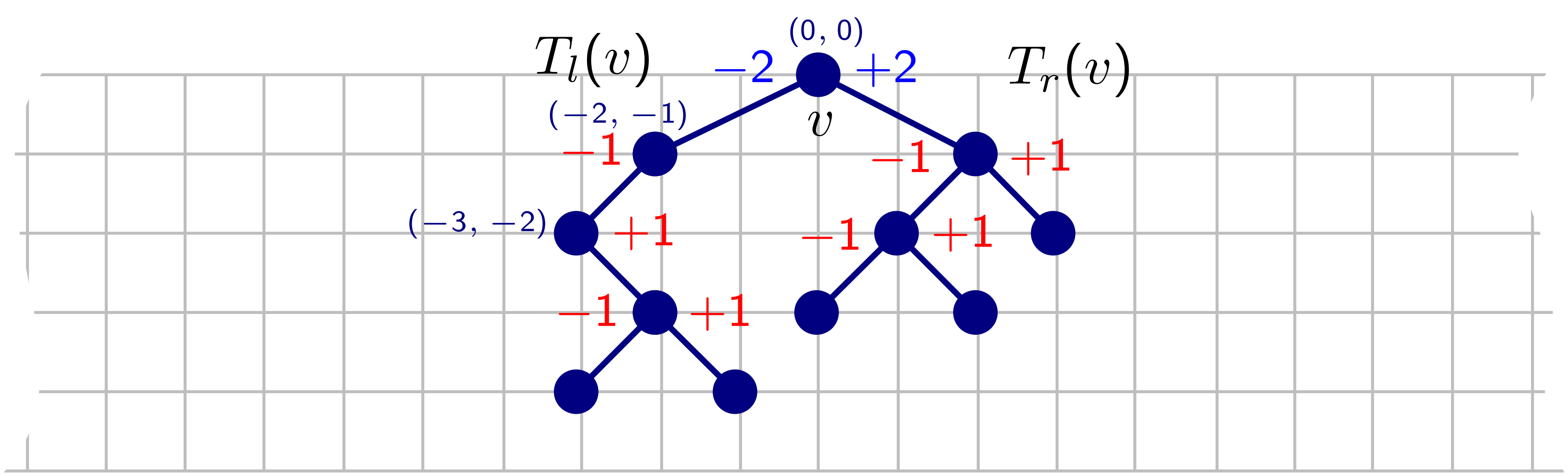


Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

Preorder traversal: Compute x- and y-coordinates.

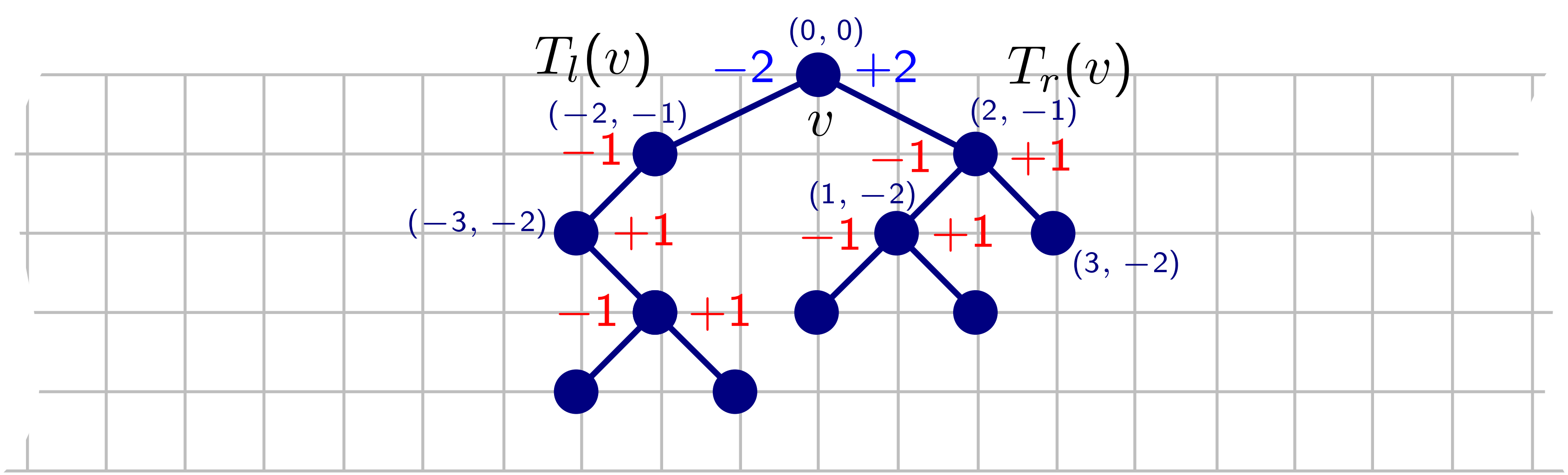


Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

Preorder traversal: Compute x- and y-coordinates.



Level-based Layout: Time Complexity

Think and write down and then discuss with your neighbour(s)

5+5 min

- What is the time complexity of the algorithm?
- What should we keep in mind to achieve this time complexity?

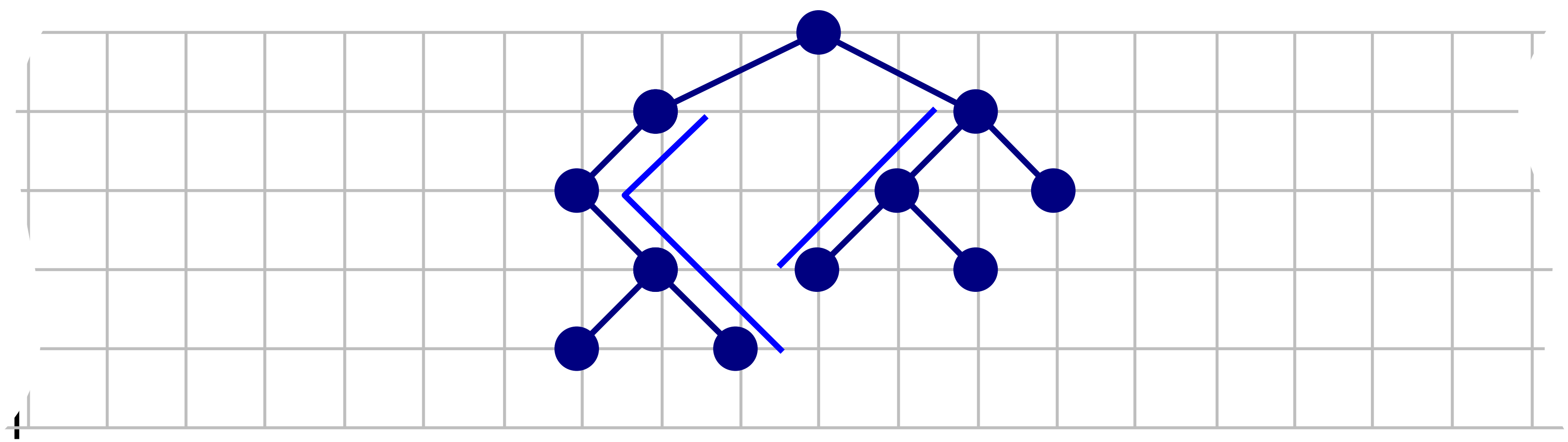
Level-based Layout

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.



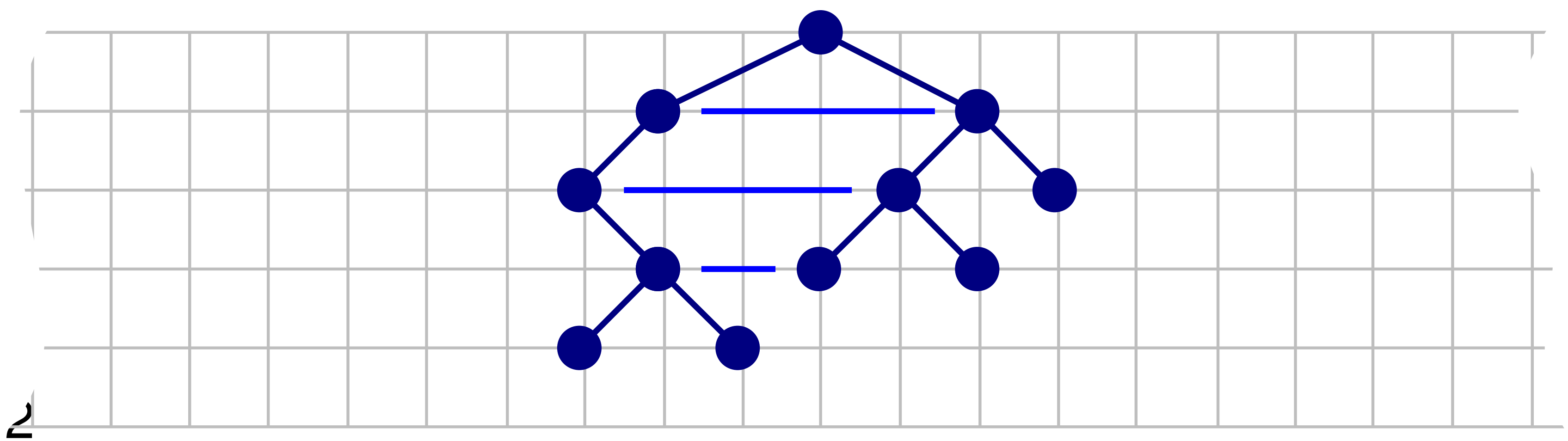
Level-based Layout

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.



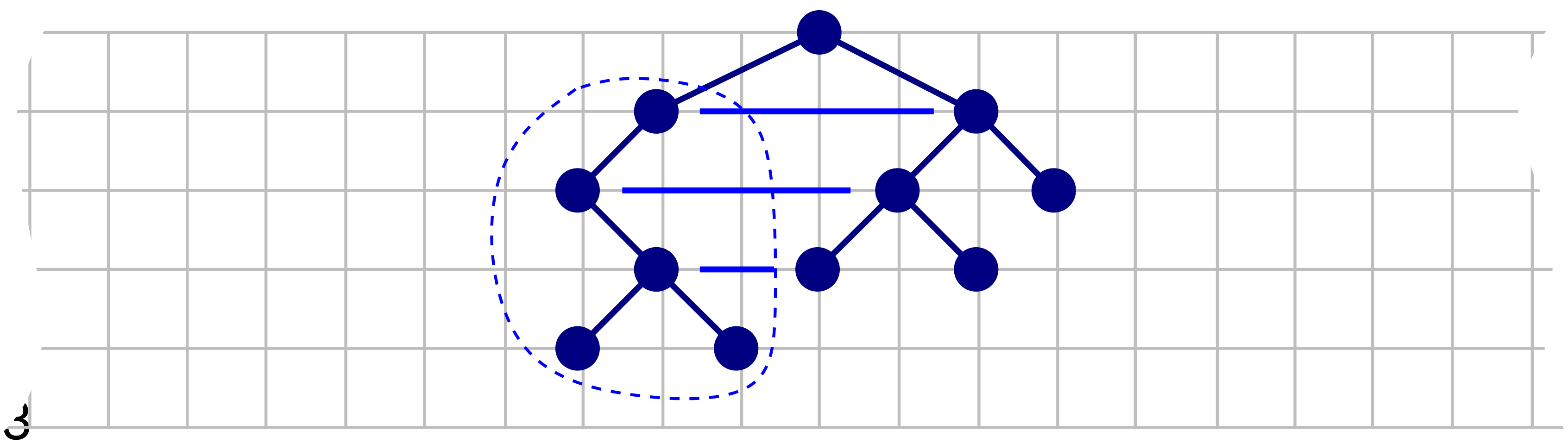
Level-based Layout

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.



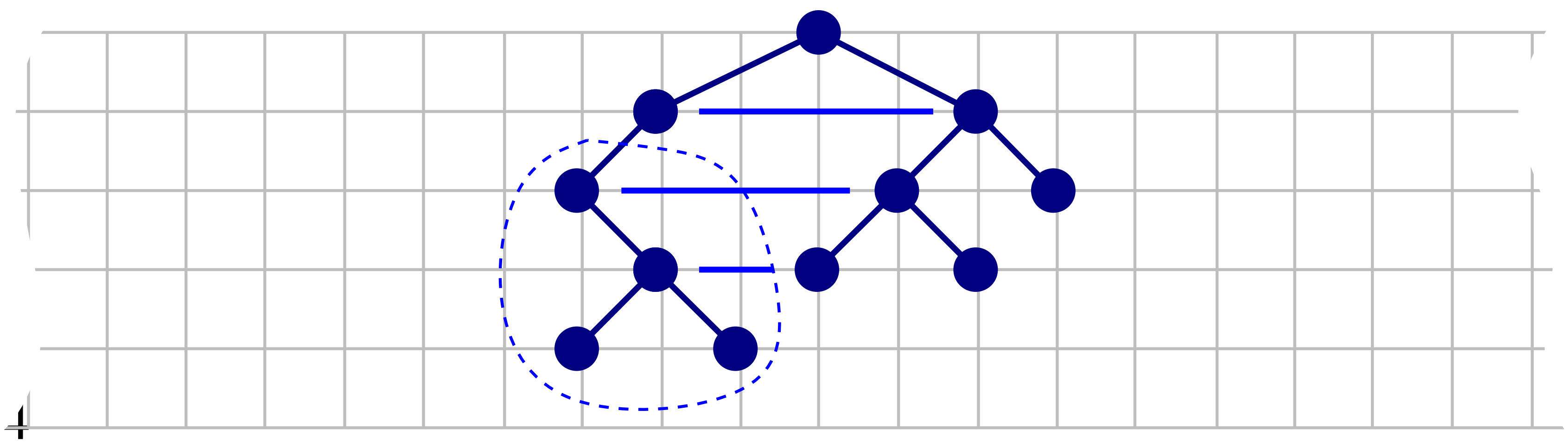
Karlsruhe Institute of Technology

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.



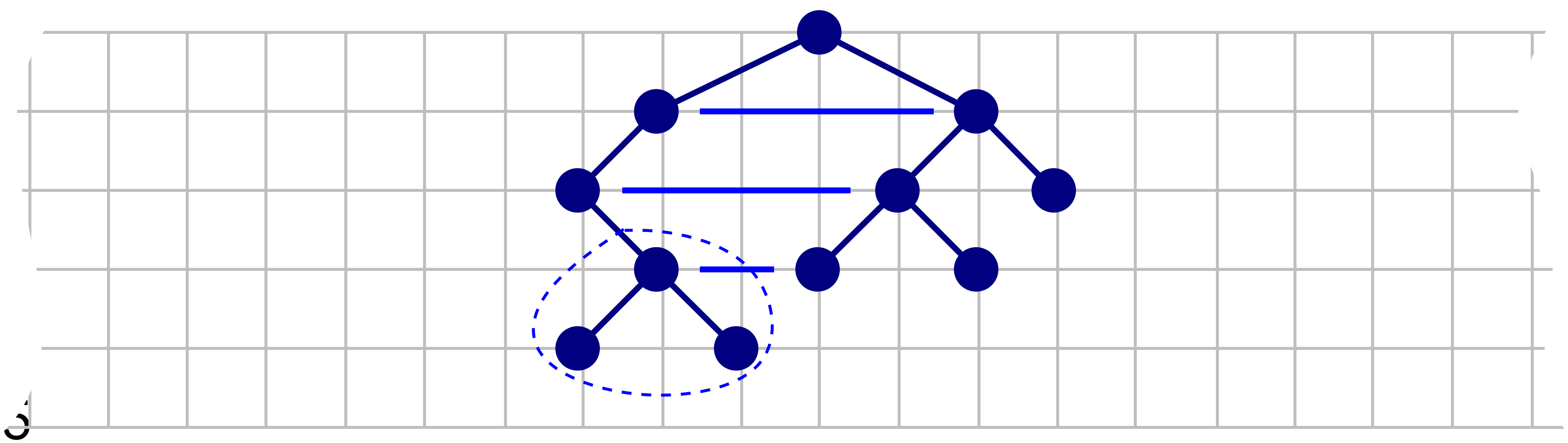
Level-based Layout

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.



Level-based Layout

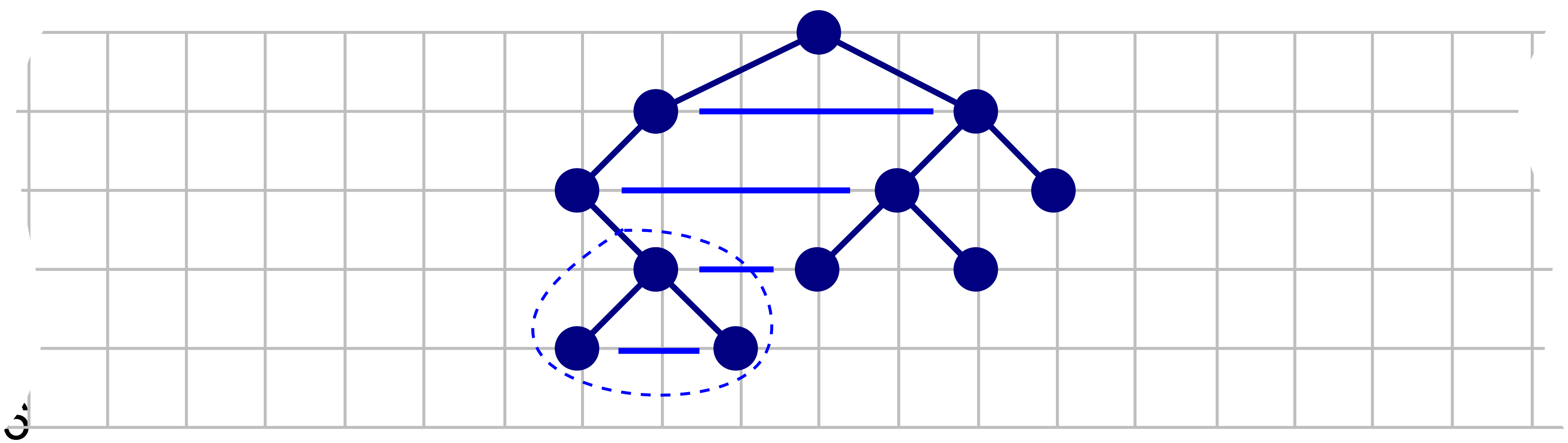
Karlsruhe Institute of Technology

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.



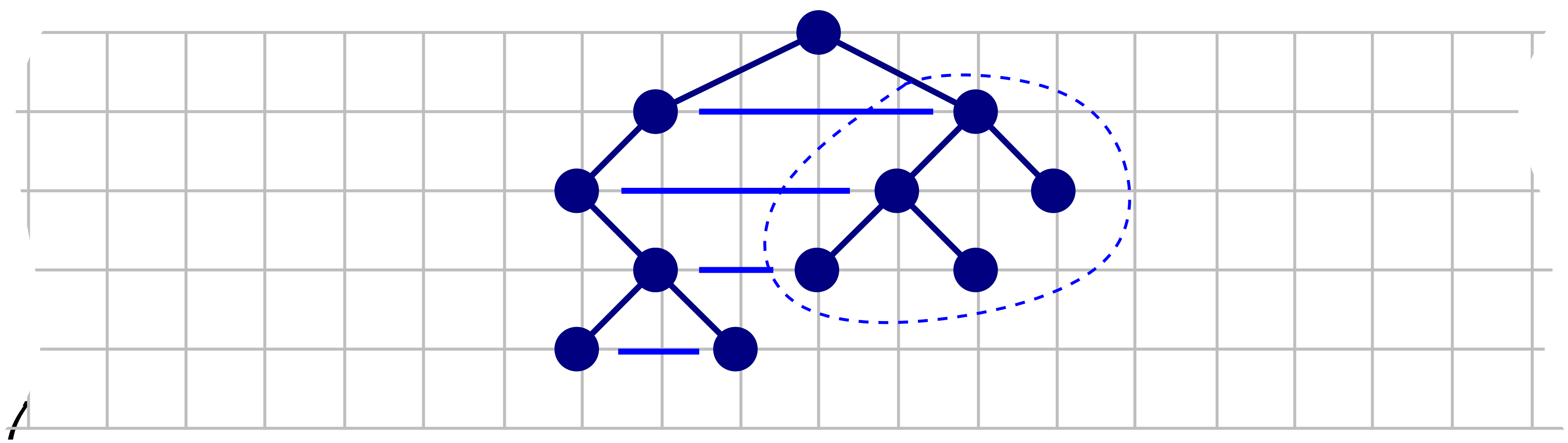
Level-based Layout

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.



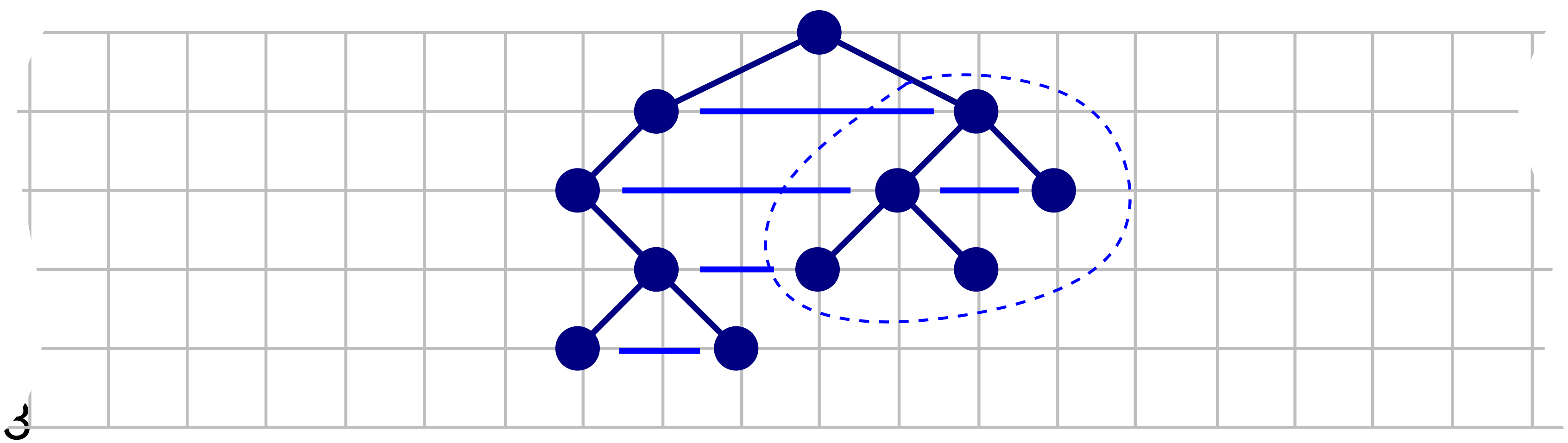
Level-based Layout

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.



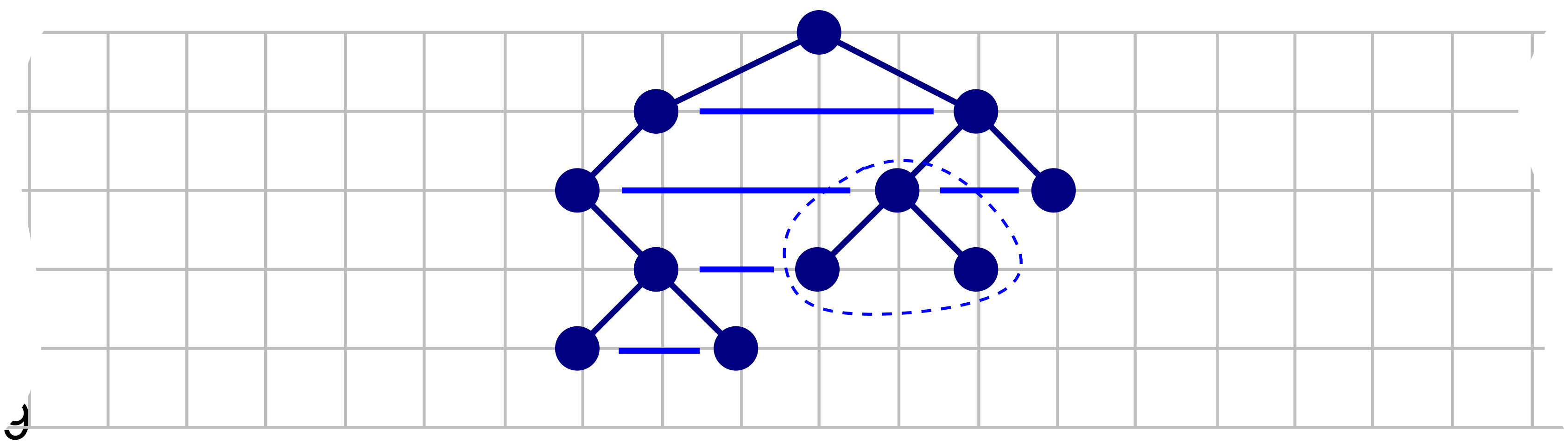
Level-based Layout

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.



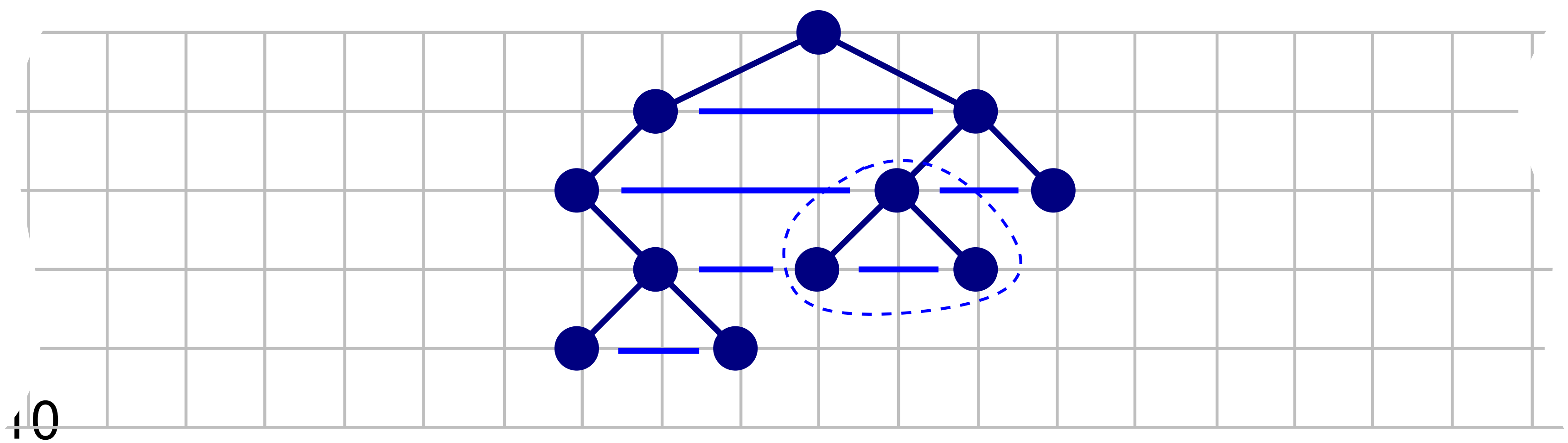
Level-based Layout

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.



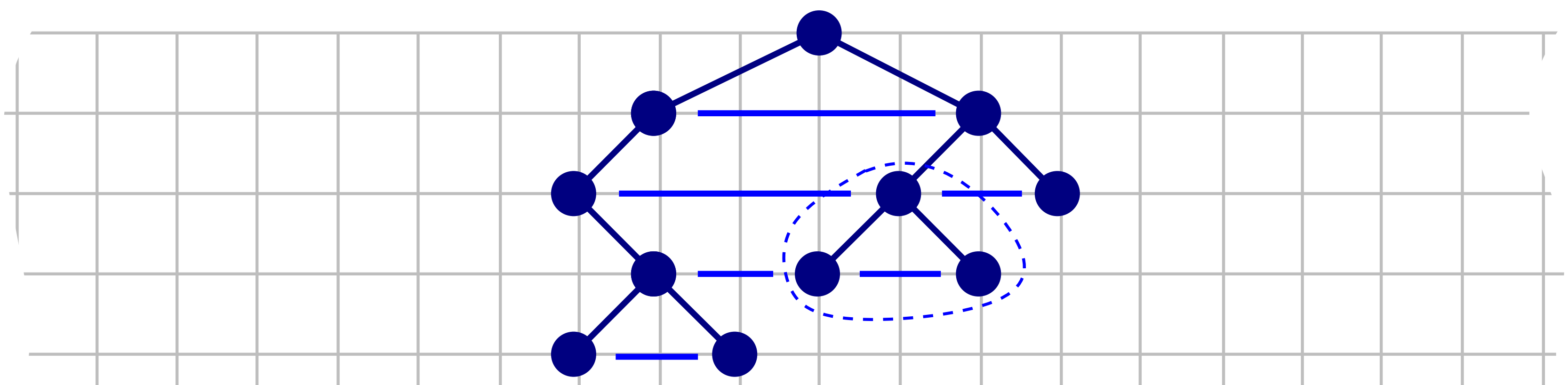
Level-based Layout

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.



- To compute the displacement: constant number of operations at each vertex

Level-based Layout

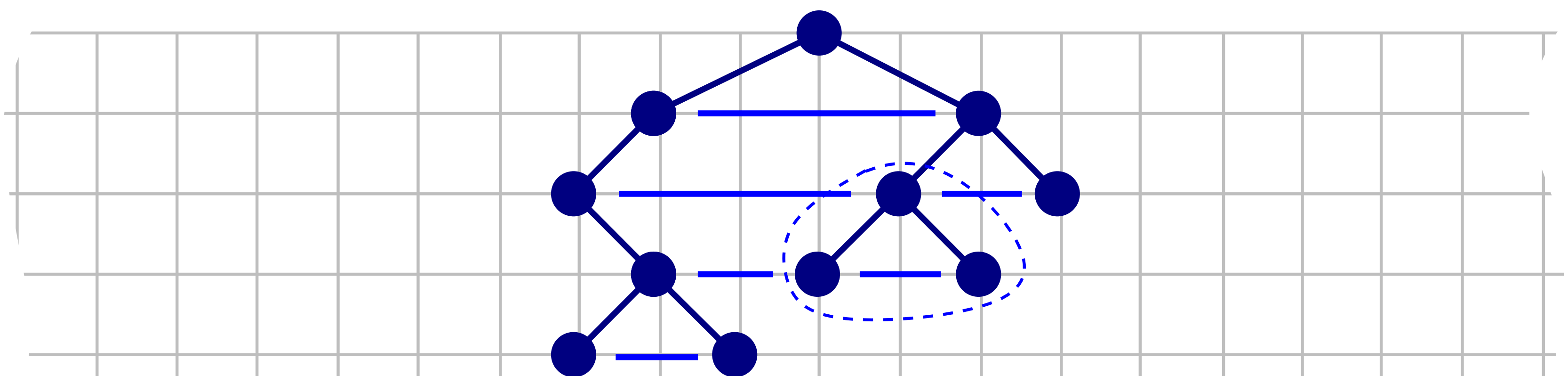
Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

$O(n)$

Preorder traversal: Compute x- and y-coordinates.



- To compute the displacement: constant number of operations at each vertex

Level-based Layout

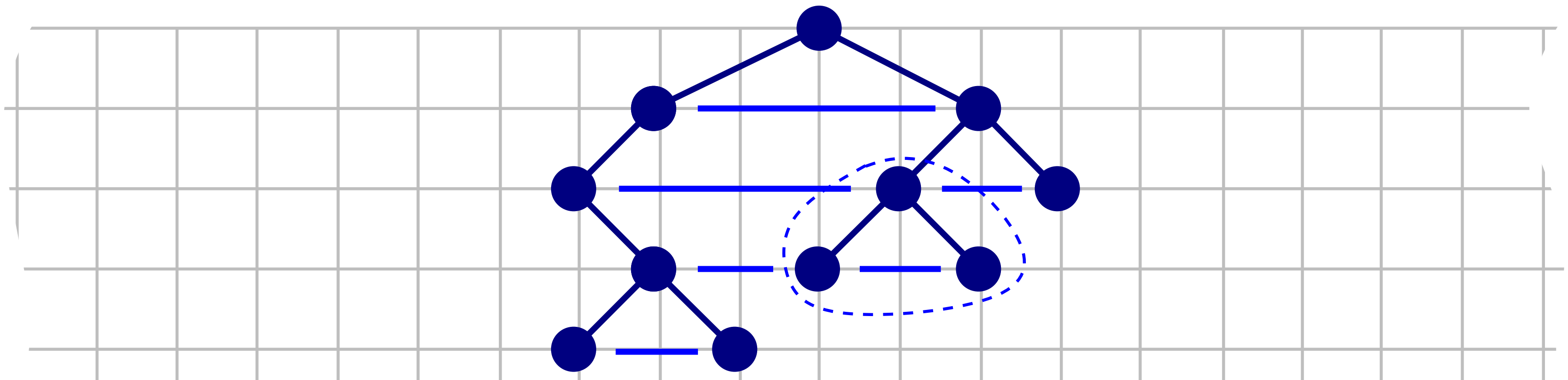
Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

$O(n)$

Preorder traversal: Compute x- and y-coordinates.



- To compute the displacement: constant number of operations at each vertex

43

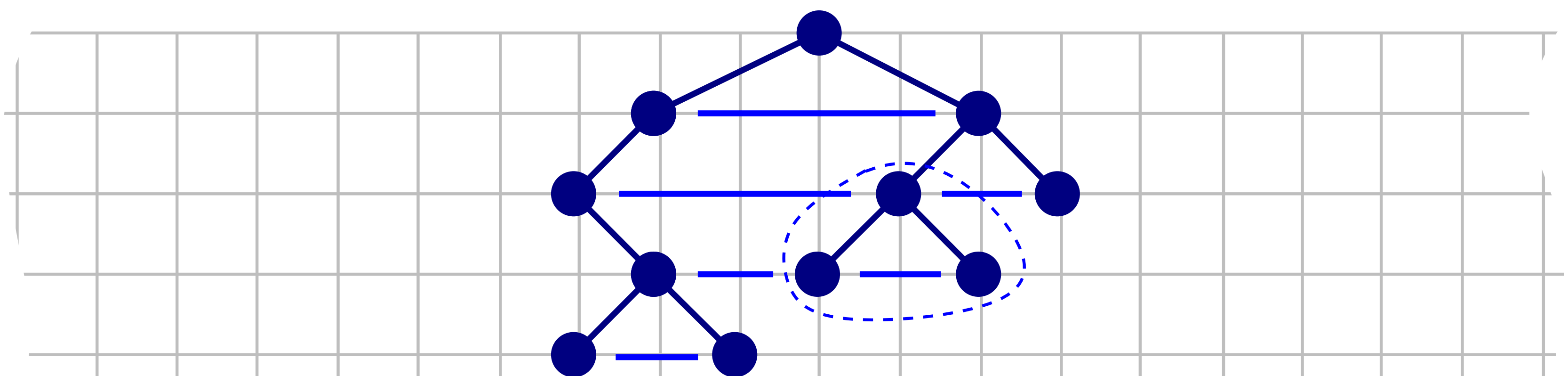
Level-based Layout

Time Complexity

Postorder traversal: For each vertex v compute horizontal displacement of the left and the right child $O(n)$

- Assume at each vertex u (below v) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at v the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates. $O(n)$



- To compute the displacement: constant number of operations at each vertex

Level-based Layout

Karlsruhe Institute of Technology

Theorem (Reingold & Tilford)

Let T be a binary tree with n vertices. Algorithm (R & T) constructs a drawing Γ of T in $O(n)$ time, such that:

- Γ is planar and straight-line
- $\forall v \in T$ y-coordinate of v is $-depth(v)$
- Vertical and horizontal distance is at least 1
- Area of Γ is

Level-based Layout

Theorem (Reingold & Tilford)

Let T be a binary tree with n vertices. Algorithm (R & T) constructs a drawing Γ of T in $O(n)$ time, such that:

- Γ is planar and straight-line
- $\forall v \in T$ y-coordinate of v is $-depth(v)$
- Vertical and horizontal distance is at least 1
- Area of Γ is $O(n^2)$

Level-based Layout

Theorem (Reingold & Tilford)

Let T be a binary tree with n vertices. Algorithm (R & T) constructs a drawing Γ of T in $O(n)$ time, such that:

- Γ is planar and straight-line
- $\forall v \in T$ y-coordinate of v is $-depth(v)$
- Vertical and horizontal distance is at least 1
- Area of Γ is $O(n^2)$
- Each vertex is centered with respect to its children

Level-based Layout

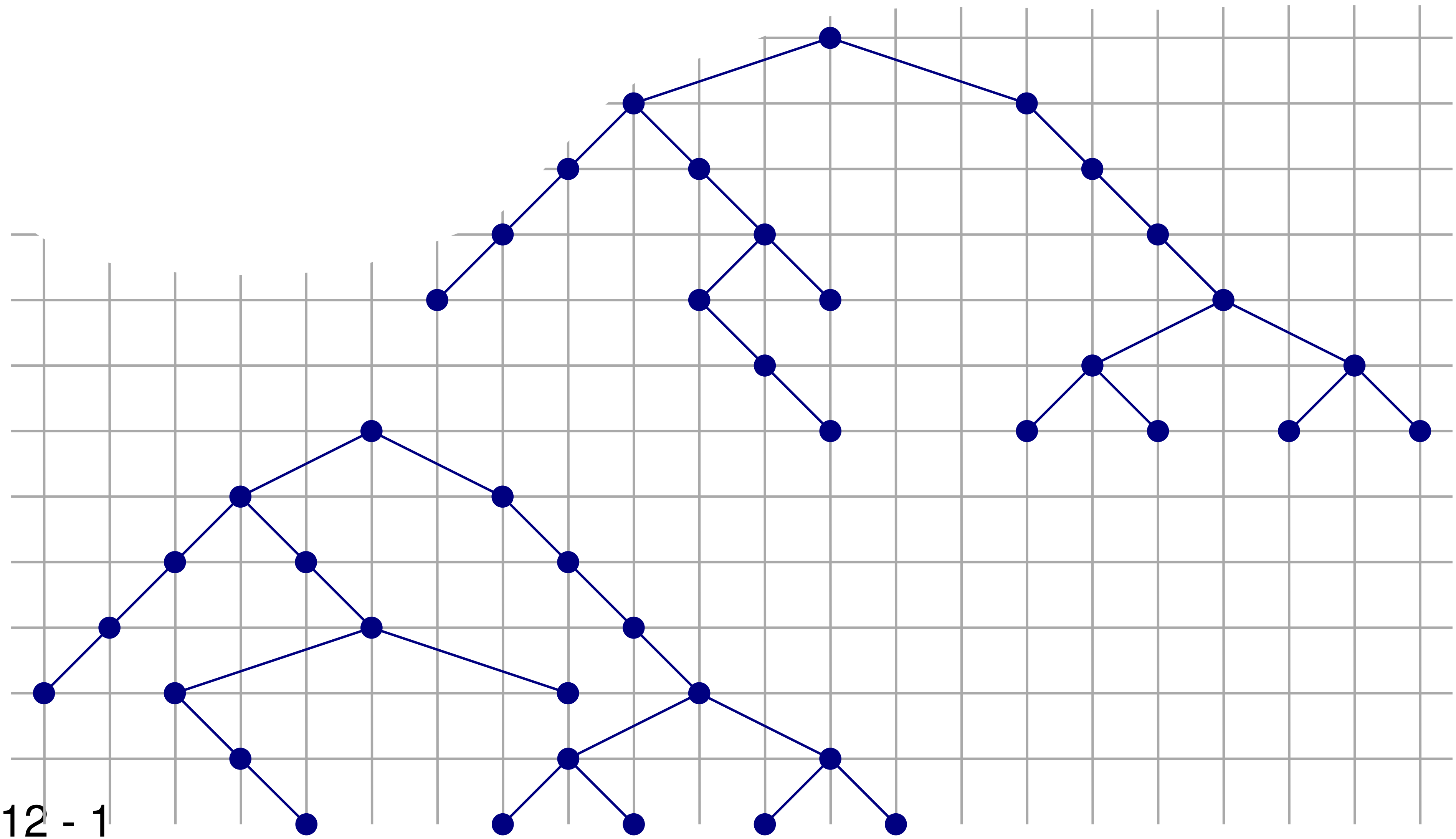
Theorem (Reingold & Tilford)

Let T be a binary tree with n vertices. Algorithm (R & T) constructs a drawing Γ of T in $O(n)$ time, such that:

- Γ is planar and straight-line
- $\forall v \in T$ y-coordinate of v is $-depth(v)$
- Vertical and horizontal distance is at least 1
- Area of Γ is $O(n^2)$
- Each vertex is centered with respect to its children
- Simply isomorphic subtrees have congruent (coincident) drawing, up to translation
- Axially isomorphic trees have congruent drawing, up to translation and reflection around y-axis

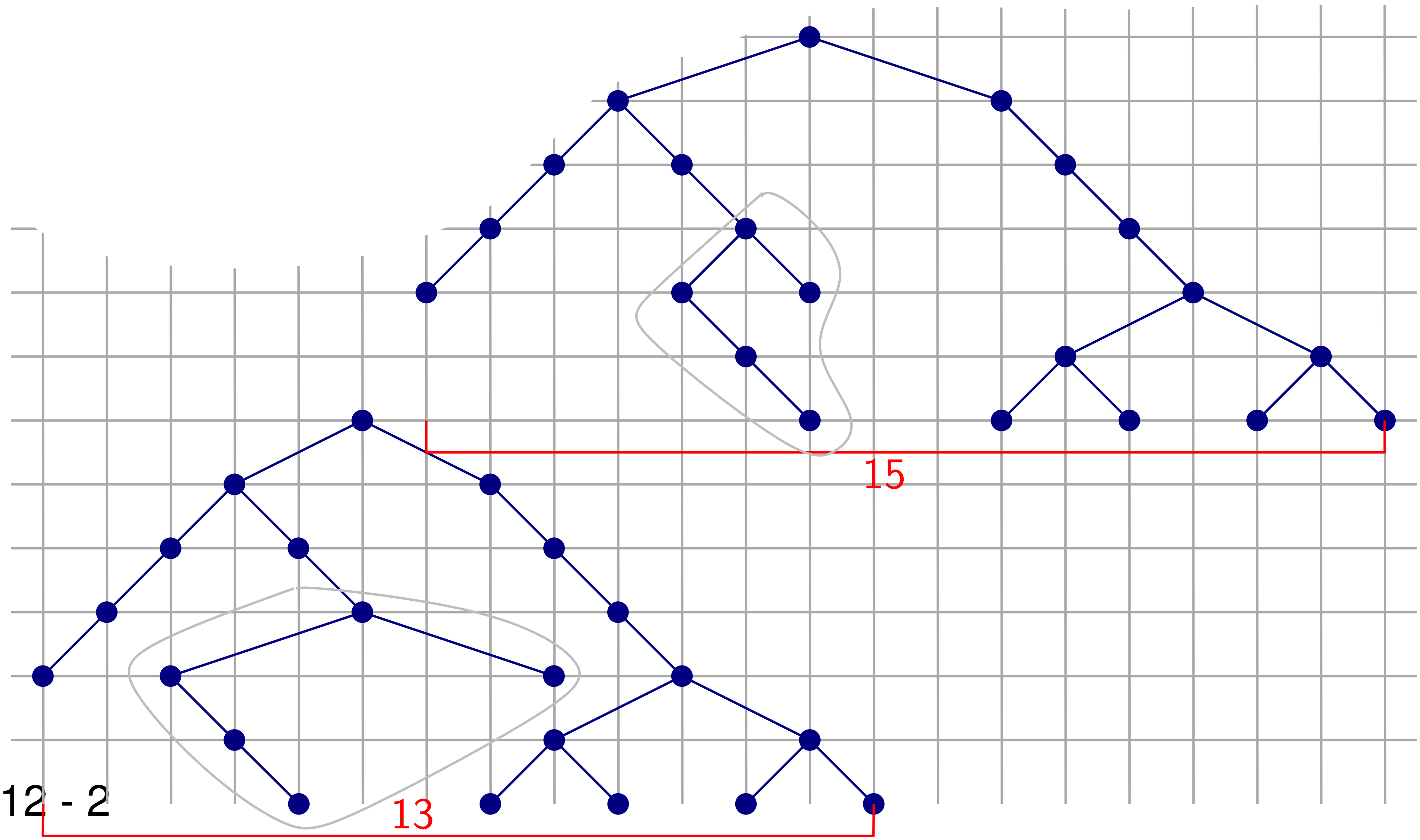
Level-based Layout

- The presented algorithm tries to minimize width



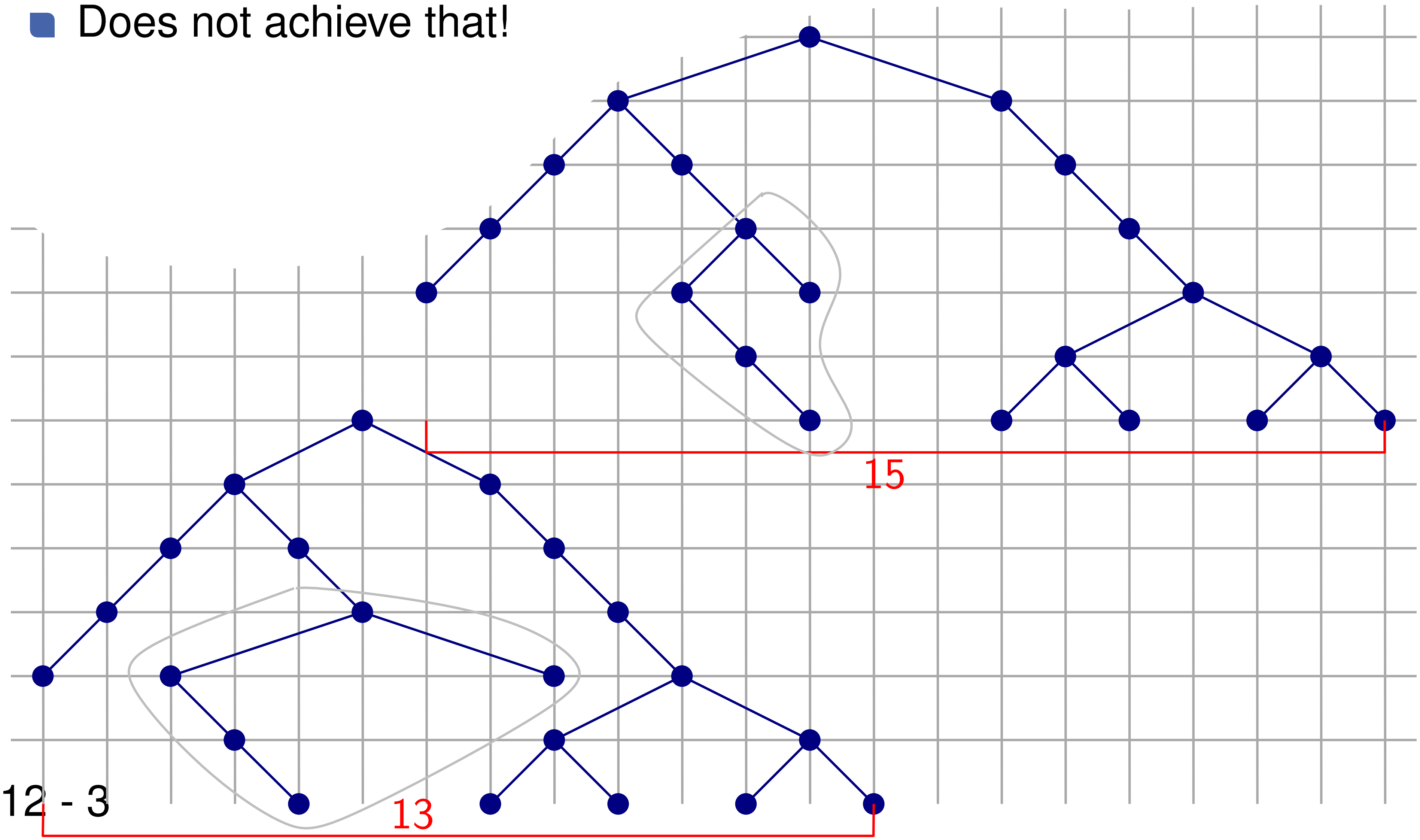
Level-based Layout

- The presented algorithm tries to minimize width



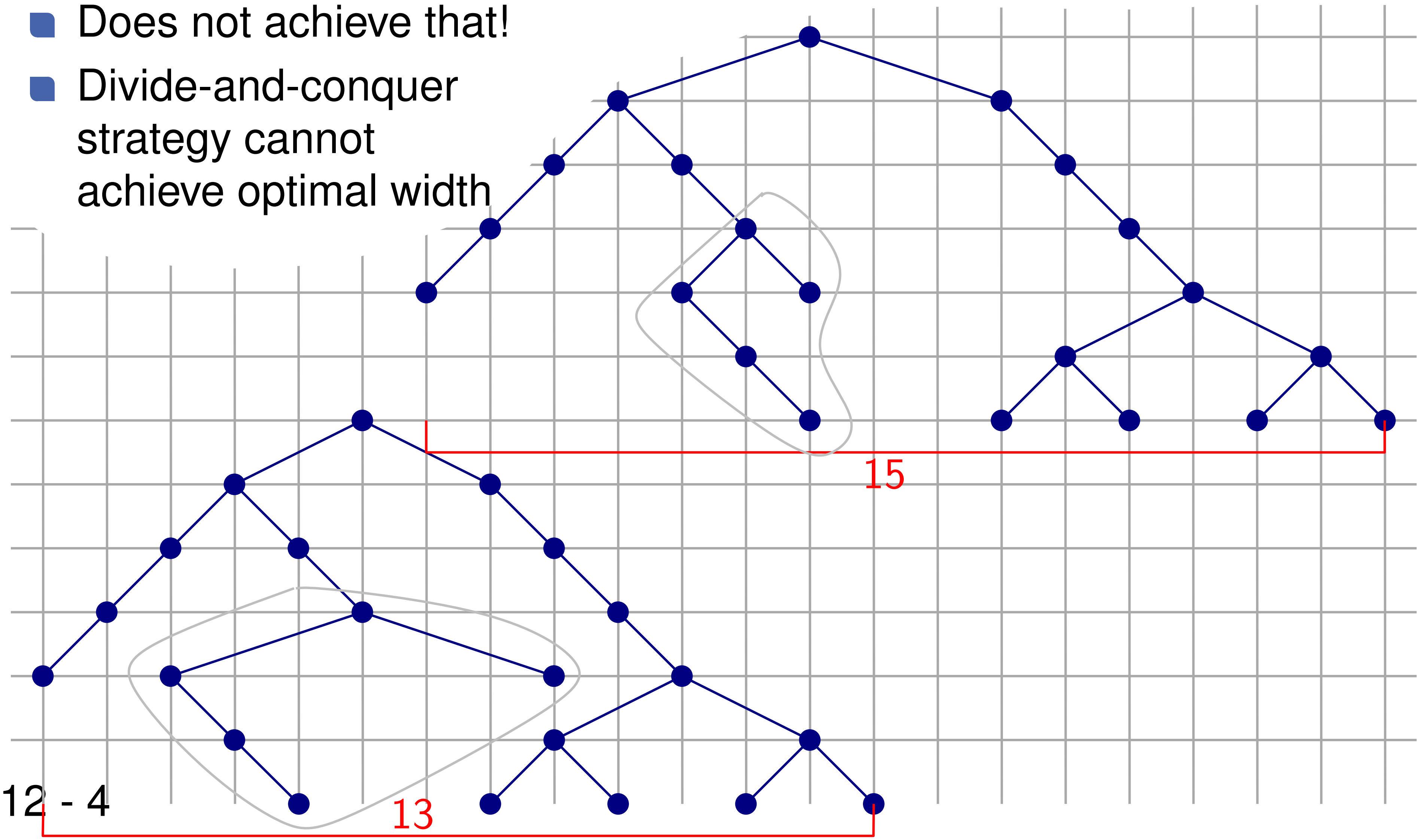
Level-based Layout

- The presented algorithm tries to minimize width
- Does not achieve that!



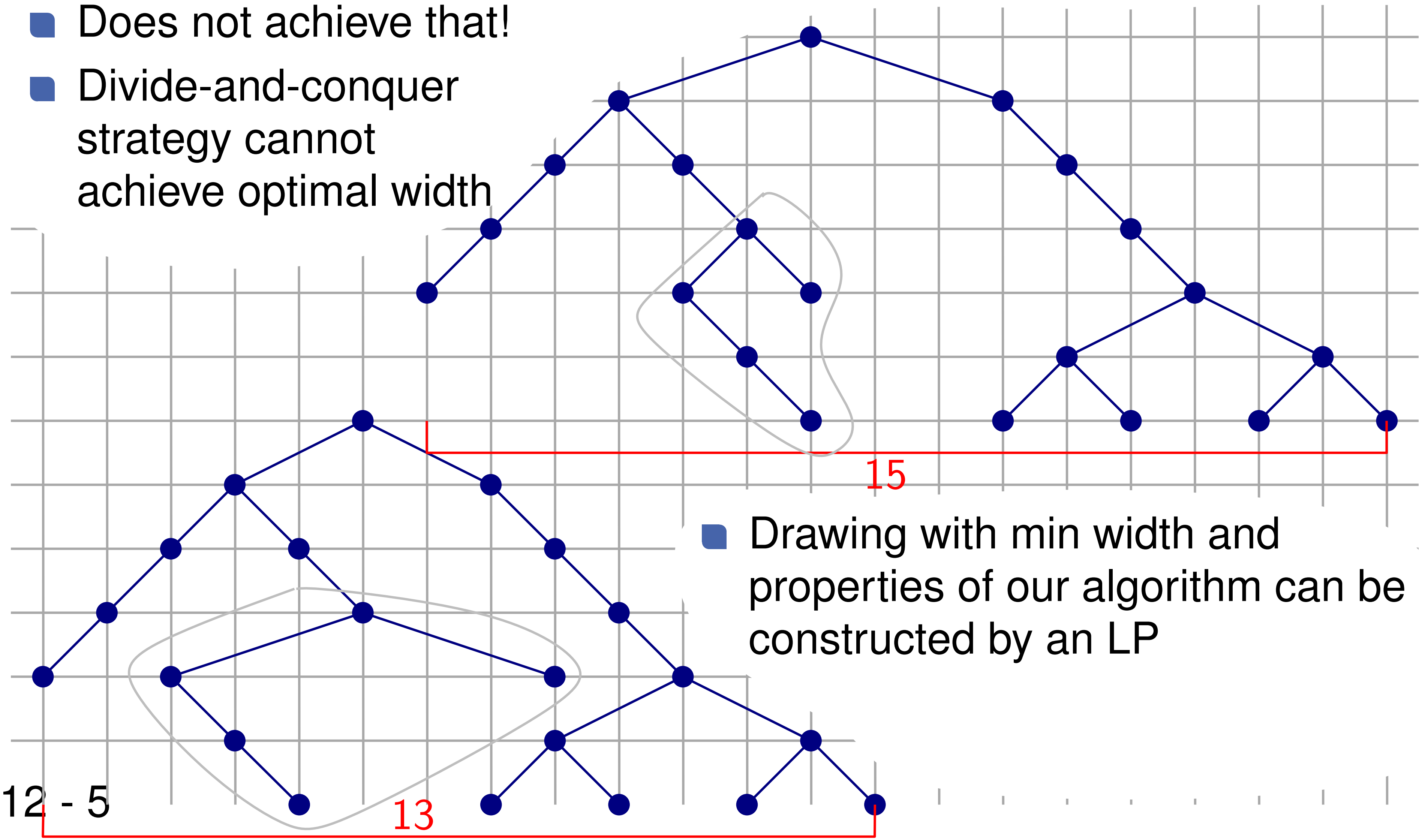
Level-based Layout

- The presented algorithm tries to minimize width
- Does not achieve that!
- Divide-and-conquer strategy cannot achieve optimal width



Level-based Layout

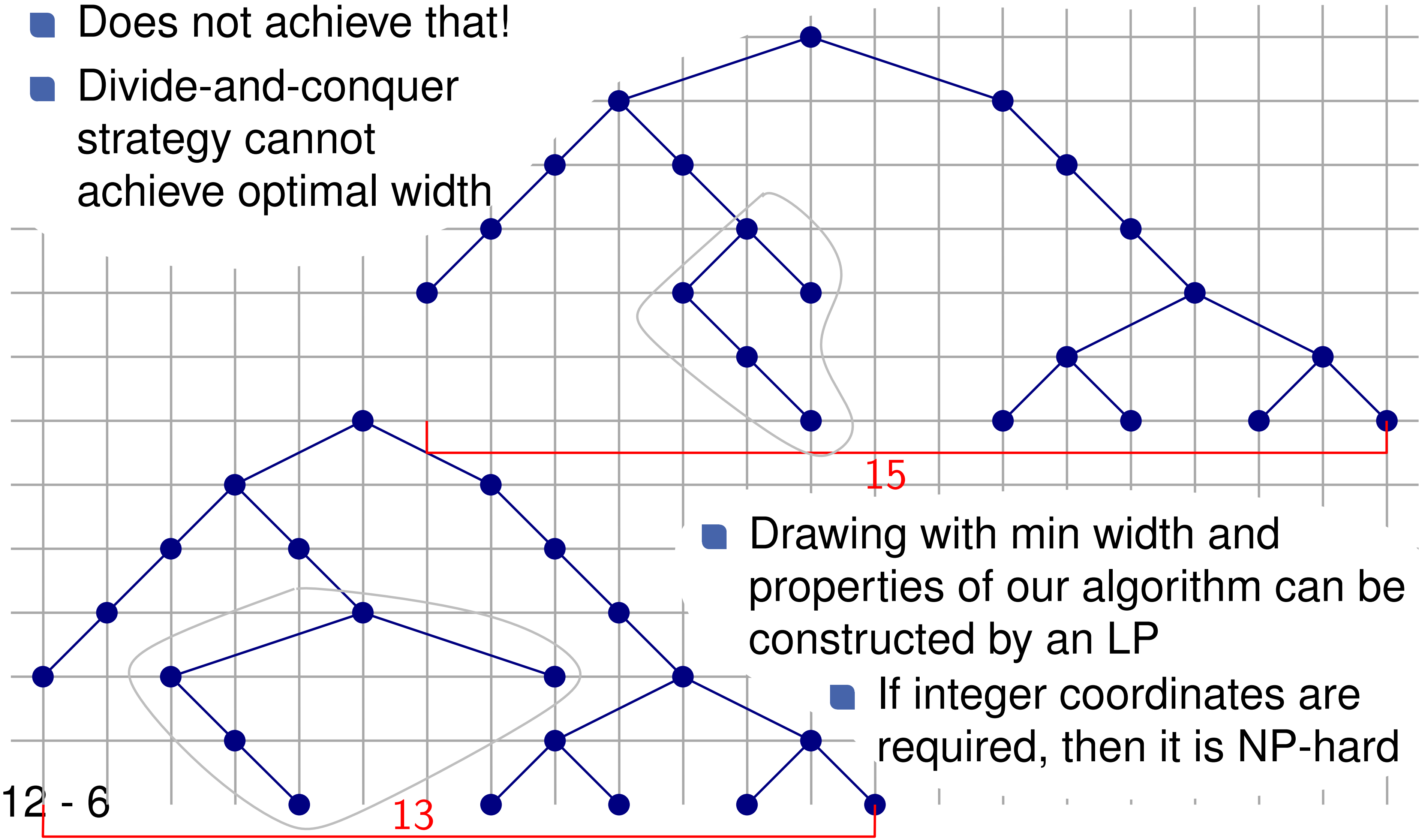
- The presented algorithm tries to minimize width
- Does not achieve that!
- Divide-and-conquer strategy cannot achieve optimal width



- Drawing with min width and properties of our algorithm can be constructed by an LP

Level-based Layout

- The presented algorithm tries to minimize width
- Does not achieve that!
- Divide-and-conquer strategy cannot achieve optimal width



- Drawing with min width and properties of our algorithm can be constructed by an LP
  - If integer coordinates are required, then it is NP-hard

Literature

Level-based Layout for Trees

- Book Di Battista et al: Chapter 3.1.2
- Skript: Chapter 6.1